Thèse

présentée par

Isabelle CRAMPÉ

pour obtenir le titre de docteur

de l'université Joseph Fourier - Grenoble 1

(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

Spécialité : INFORMATIQUE

Révision interactive dans une base de connaissance à objets

Date de soutenance : 14 octobre 1997

Composition du jury : Roland Ducournau (rapporteur)
Jérôme Euzenat (directeur)
Philippe Jorrand (examinateur)
Odile Papini (rapporteur)
Laurent Trilling (examinateur)

Thèse préparée au sein de l'INRIA Rhône-Alpes

Je tiens à remercier …

Madame Odile Papini, maître de conférence à l'université d'Aix en Provence, pour avoir accepté d'être rapporteur de ma thèse et pour les remarques faites sur mon travail.

Monsieur Roland Ducournau, professeur à l'université de Montpellier, qui a bien voulu être rapporteur de ma thèse. Les discussions régulières au cours de ces trois dernières années ont amené critiques et remarques très enrichissantes.

Monsieur Philippe Jorrand, professeur à l'INPG, pour avoir accepté d'examiner mon travail et de faire partie du jury de ma thèse.

Monsieur Laurent Trilling, professeur à l'université Joseph Fourier de Grenoble, qui s'est penché sur mon travail de thèse et accepté d'en être examinateur.

Jérôme Euzenat, chargé de recherche INRIA, qui m'a guidée, soutenue, relue, éclairée, corrigée, supportée (et j'en oublie sûrement) durant les quatre années de mon DEA et de ma thèse. Je le remercie pour sa disponibilité et sa collaboration des plus fructueuses.

Je tiens aussi à remercier toutes les personnes que j'ai côtoyées durant ces dernières années :

les membres du projet Sherpa : François Rechenmann qui m'a acceptée dans son équipe ; Françoise De Coninck toujours disponible pour toutes les tracasseries administratives ; Patrice et Pierre qui m'ont bien souvent dépannée ; Sueli, Jean-Marc, Thibault, Cécile, Florence, Petko, Karine, qui ont partagé ma condition de thésard et aidé à garder le sourire ; Danièle, Gilles, Michel, Jérôme, Sylvie, pour toutes nos discussions ; et tous ceux qui ont fait irruption dans le projet : Loïc, les Christophes, Vincent, …

les amateurs quotidiens de la pause thé : Catherine, Sueli, Véronique, Thibault, Loïc, Anne, pour les échanges sur la recherche et tout le reste …

des enseignants de l'université Pierre Mendès-France : Georges, Jean-Michel, Bernard, Luc, Agnès, François, que j'ai côtoyés avec plaisir dans mes activités d'enseignement.

Et enfin, ma famille pour son soutien à distance, et tous mes amis, qui ont fait de ces années une période de ma vie très riche en rencontres, balades, partages et fêtes : Catherine bien sûr, mais aussi Claire, Nicolas, Cécile, Fred, Caro, Benoît, Marc, Florence, Denis, Claude & Denis et tous les charmants chefs scouts, tous ceux que j'ai été amenée à rencontrer plus ou moins tardivement, … sans oublier Lionel ! A tous, merci et bon vent …

Résumé

Lorsqu'une base de connaissance est construite de manière incrémentale, le dernier ajout peut être contradictoire avec le contenu de la base. Or, l'objectif d'une base de connaissance est de modéliser un domaine et elle doit donc être consistante, c'est-à-dire admettre au moins un modèle. Pour ajouter une connaissance inconsistante avec la base, il faut donc modifier celle-ci afin de préserver sa consistance. Cette problématique se rapproche de celle de la révision dans les langages logiques, dont le principal inconvénient est la complexité qui ne permet pas l'implémentation.

L'objectif est de définir une révision, dans le cadre des représentations de connaissance par objets, qui puisse être implémentée, notamment en tenant compte des particularités des langages de représentation par objets.

Dans un premier temps, nous définissons formellement un langage d'objets : sa sémantique et un système déductif syntaxique correct et complet par rapport à la sémantique. De plus, nous définissons syntaxiquement l'inconsistance, ce qui permet de la détecter en se basant sur les propriétés de localité du langage. Contrairement aux langages logiques classiques, une inconsistance ne permet pas de tout déduire et reste donc localisée.

Dans un second temps, nous définissons les bases révisées qui satisfont les postulats classiques de la révision, en particulier la minimalité, principe selon lequel il faut perdre le moins possible de connaissance. La minimalité peut s'interpréter intuitivement selon la relation d'ordre entre les classes. Cependant, elle est basée sur l'inclusion ensembliste et n'est pas un critère suffisant pour obtenir une unique base. Un algorithme, qui a été implémenté, propose donc toutes les bases révisées minimales suite à un ajout inconsistant ; il est interactif afin de maîtriser la complexité inhérente à la révision.

Mots-clefs : révision, représentation de connaissance par objets, inconsistance, raisonnement non monotone.

Abstract

Since knowledge on a domain is always evolving, knowledge bases must be incrementally built. A new piece of knowledge may be contradictory with older ones but base consistency must be preserved, i.e. there is still a model. This consistency is necessary since a base has to represent a domain. This problem is near revision in logic, but in logic language the trouble is that programming is impossible due to complexity.

The aim is to define revision for object-based representation, in order to implement an efficient algorithm, by using object features.

First, an object-based language is defined : its semantics and a complete and sound set of deduction rules. Then, inconsistency is syntactically characterised, and detection takes advantage of the fact that inconsistency does not spread in the whole object-based base, since it does not allow to deduce everything.

Then, a definition of revised bases, i.e. parts of initial base that are consistent with the new piece of knowledge, is proposed. This revision satisfies classical logic postulates and in particular, minimality, that is, the least knowledge must be lost. Minimality is shown to have an intuitive meaning according to the class specialisation. Whenever several revised bases are possible, the choice is assumed to be made by the user. Eventually, we describe an implemented algorithm which gives all the possible minimal modifications of a base inconsistent with a new piece of knowledge. Such an interactive revision in object context allows to avoid main classical revision problems, and in particular can be implemented by efficient algorithms.

Sommaire

INTRODUCTION **5**

CHAPITRE I. REPRÉSENTATION DE CONNAISSANCE PAR OBJETS ET RÉVISION. **9**

A. REPRÉSENTATION DE CONNAISSANCE 10

B. REPRÉSENTATION PAR OBJETS 10

C. TROPES 14

D. LANGAGE TERMINOLOGIQUE 17

E. LA CONSTRUCTION D'UNE BASE 18

F. COMMENT RÉVISER UNE BASE ? 30

G. QUELLE RÉVISION DANS LES REPRÉSENTATIONS PAR OBJETS ? 42

CHAPITRE II. FORMALISATION D'UN LANGAGE D'OBJETS **49**

A. UN LANGAGE D'OBJETS SIMPLE 50

B. LANGAGE AVEC RÉFÉRENCE AUX INSTANCES 71

C. COMPARAISON AVEC D'AUTRES APPROCHES 81

D. CONCLUSION 83

CHAPITRE III. RÉVISION DANS UN LANGAGE D'OBJETS **85**

A. INCONSISTANCE ET INCOHÉRENCE 85

B. BASE RÉVISÉE 95

C. MINIMALITÉ 99

D. PRINCIPES SATISFAITS 103

E. SYNTHÈSE DES PRINCIPALES ORIGINALITÉS ET CARACTÉRISTIQUES DE LA RÉVISION PROPOSÉE 111

CHAPITRE IV. ALGORITHME ET EXPÉRIMENTATION **113**

A. GÉNÉRALITÉS SUR L'ALGORITHME 113

B. ALGORITHME SUR LE LANGAGE SIMPLE 117

C. ALGORITHME SUR LE LANGAGE AVEC RÉFÉRENCE AUX INSTANCES 134

D. DES ALTERNATIVES À LA RÉVISION 157

E. TROPES EXPRIMÉ DANS LE LANGAGE FORMEL 162

F. EXEMPLES DE RÉVISION 166

G. DES FONCTIONNALITÉS SUPPLÉMENTAIRES 179

CONCLUSION **181**

PERSPECTIVES **183**

ANNEXES **186**

BIBLIOGRAPHIE **260**

Introduction

Une base de connaissance permet de représenter ce qui est connu sur un domaine particulier afin de pouvoir structurer les informations et inférer de nouvelles connaissances. Il peut s'agir aussi bien des gènes d'un organisme en biologie moléculaire que de l'organisation informatique d'une entreprise. Une telle base ne peut généralement pas être construite de manière définitive car le domaine peut évoluer, et, même si le domaine est stable, les connaissances sur le domaine évoluent, au fil des expérimentations ou des nouvelles modélisations. Dans l'hypothèse d'un domaine non évolutif, la construction est donc réalisée par des ajouts successifs afin que la base modélise le domaine de la manière la plus fiable possible.

Généralement, la construction de la base se fonde sur des expériences et des modélisations qui proviennent de différentes sources et qui ne sont pas complètement fiables, donc la cohérence de l'ensemble des connaissances n'est pas assurée. Or, pour pouvoir utiliser une base de connaissance, il est nécessaire qu'elle ne recèle pas de contradiction, sous peine de pouvoir déduire une connaissance et son contraire. Prenons l'exemple simpliste d'une base contenant les connaissances suivantes : "Les mammifères n'ont pas de bec" et "Piwi est un mammifère". Si une expérience révèle que "Piwi a un bec", car c'est un ornithorynque, l'ajout de cette connaissance permet de déduire à la fois que "Piwi a un bec" et que "Piwi n'a pas de bec".

Or, il n'est pas toujours possible pour un concepteur de base de connaissance d'avoir une vision globale du contenu de la base, qui peut être de taille importante, car la construction d'une base, un tant soit peu complexe, s'étale dans le temps et qu'elle requiert souvent plusieurs personnes. Ainsi, les concepteurs ne se rendent pas compte qu'un ajout est contradictoire avec le contenu de la base si la contradiction met en jeu des connaissances d'une partie de la base qui leur est inconnue. Il est alors important, tout d'abord, de détecter automatiquement les contradictions afin de pouvoir assurer la cohérence de l'ensemble des connaissances contenues dans la base. Si la dernière connaissance à ajouter – celle qui a déclenché la contradiction – n'est finalement pas jugée primordiale par le concepteur, il lui suffit de ne pas l'inclure dans la base. Dans l'exemple précédent, si la dernière expérience n'est pas prise en compte, Piwi sera alors toujours considéré comme n'ayant pas de bec. Au contraire, la dernière connaissance introduite peut être celle dans laquelle on a le plus confiance, car les conclusions de l'expérience sont jugées irréfutables par le concepteur ou que les hypothèses sous-jacentes à une partie de la base ne sont plus valides. Il s'agit donc d'en

tenir compte et de l'insérer dans la base tout en préservant la cohérence ; pour cela, certaines des connaissances précédentes, celles qui sont en contradiction avec la dernière introduite, doivent obligatoirement être supprimées. Il existe généralement plusieurs sous-ensembles de connaissances dont la suppression permet l'ajout de la dernière dans la base tout en préservant la cohérence. Ainsi, afin de lever la contradiction sur le bec de Piwi, il est possible de décider par exemple que les mammifères peuvent avoir un bec, ou que Piwi n'est pas un mammifère. Cependant, il suffit que la base soit volumineuse ou mal documentée pour que la recherche manuelle des connaissances à supprimer devienne très laborieuse, car il faut extraire des sous-ensembles de connaissances de la base et vérifier que leur suppression permet de garder la base cohérente.

Il est donc utile de disposer d'un mécanisme qui propose au concepteur les possibilités de suppressions dans la base afin de pouvoir y insérer la dernière connaissance obtenue sans produire de contradiction. Cette problématique se rapproche fortement de la définition de la révision dans les systèmes logiques. La révision est définie généralement par le biais d'un opérateur qui prend pour arguments un ensemble de propositions et une proposition inconsistante avec l'ensemble ; il fournit un ensemble consistant contenant la dernière proposition et le maximum de propositions de l'ensemble. L'un des impératifs de la révision est en effet de satisfaire le principe de minimalité qui consiste à conserver le maximum de connaissances de la base : comme la nouvelle connaissance est inconsistante avec le reste de la base, la base est erronée, mais il est vraisemblable qu'elle représente quand même relativement bien le domaine. Cette notion de minimalité a diverses acceptions, en particulier selon que les connaissances ont toutes la même importance ou non et selon que l'on considère le nombre de connaissances modifiées ou l'inclusion ensembliste des bases après révision. Dans notre cadre, la minimalité sera également déterminante pour trouver les suppressions nécessaires à la préservation de la consistance. Ainsi, toujours dans l'exemple de Piwi, la suppression de "Les mammifères n'ont pas de bec" *et* de "Piwi est un mammifère" conduit à la base contenant "Piwi a un bec". Cette base est consistante, mais elle ne correspond manifestement pas à une révision minimale car la suppression de *l'une des deux* connaissances ("Les mammifères n'ont pas de bec" ou "Piwi est un mammifère") suffit à obtenir une base consistante. Par contre, il n'est pas possible de choisir – sans plus d'information – entre la suppression de "Les mammifères n'ont pas de bec" ou de "Piwi est un mammifère", donc les bases correspondant à ces deux suppressions sont considérées comme des solutions.

L'objectif de ce travail est de proposer un mécanisme de révision pour les systèmes de représentation de connaissance par objets. Les bases exprimées dans ces langages peuvent se décrire brièvement comme contenant de la connaissance sous forme d'objets qui sont des entités décrites par leurs caractéristiques. Les objets ayant des traits

communs sont regroupés au sein de classes, qui sont ordonnées selon une relation de spécialisation qui permet de décrire les sous-classes de chaque classe : les objets contenus dans une sous-classe appartiennent également à la classe.

Notre travail se ramène à la définition des contradictions qui peuvent apparaître dans une base et à l'élaboration des différentes solutions minimales possibles parmi lesquelles le concepteur pourra choisir. Plusieurs solutions sont proposées car, pour n'en fournir qu'une seule, il aurait fallu demander aux concepteurs de la base d'établir explicitement un ordre entre les connaissances, ce qui n'est pas faisable en pratique. Pour obtenir ces solutions, la traduction des bases de connaissance à objets dans un formalisme logique n'est pas satisfaisante puisque la révision proposée pour les langages logiques est trop coûteuse pour être implémentée. Il s'agit donc de définir et d'implémenter la révision dans le cadre des représentations de connaissance par objets en tenant compte de leurs particularités, telles que la relation de spécialisation entre classes et les liens entre objets, afin d'éviter une trop grande complexité. L'interactivité avec le concepteur permet de limiter la complexité, comme c'est toujours le cas, mais conduit surtout à l'obtention de la solution la plus satisfaisante pour lui, celle qui permet de perdre le moins de connaissances primordiales à ses yeux.

Une présentation des systèmes de représentation de connaissance par objets, ainsi qu'un tour d'horizon de la gestion des inconsistances lors de la construction d'une base de connaissance et de la révision dans les langages logiques, permettent de définir précisément la problématique de la révision dans les bases de connaissance à objets (chapitre I).

Notre travail se décompose alors en trois parties :

Nous décrivons la syntaxe et la sémantique d'un langage de représentation de connaissance par objets. De plus, afin de pouvoir manipuler les bases de connaissance, il faut en donner une représentation finie (chapitre II).

Nous caractérisons ce qu'est une inconsistance dans le cadre des langages d'objets et nous définissons le mécanisme de révision : quels sous-ensembles de la base proposer au concepteur ? Comment définir le critère de minimalité ? (chapitre III)

Nous élaborons un algorithme qui calcule les bases correspondant à la révision minimale de la base initiale et de la dernière connaissance inconsistante avec elle (chapitre IV).

Les notations introduites, la liste des définitions, les preuves des propositions et l'évaluation de la complexité sont données en annexe.

Chapitre I. Représentation de connaissance par objets et révision

Ce chapitre a pour objectif d'une part de présenter la représentation de connaissance par objets et d'autre part de faire un tour d'horizon des travaux en relation avec la révision. La trame générale est illustrée sur la Figure 1, les sections concernées étant indiquées entre parenthèses.

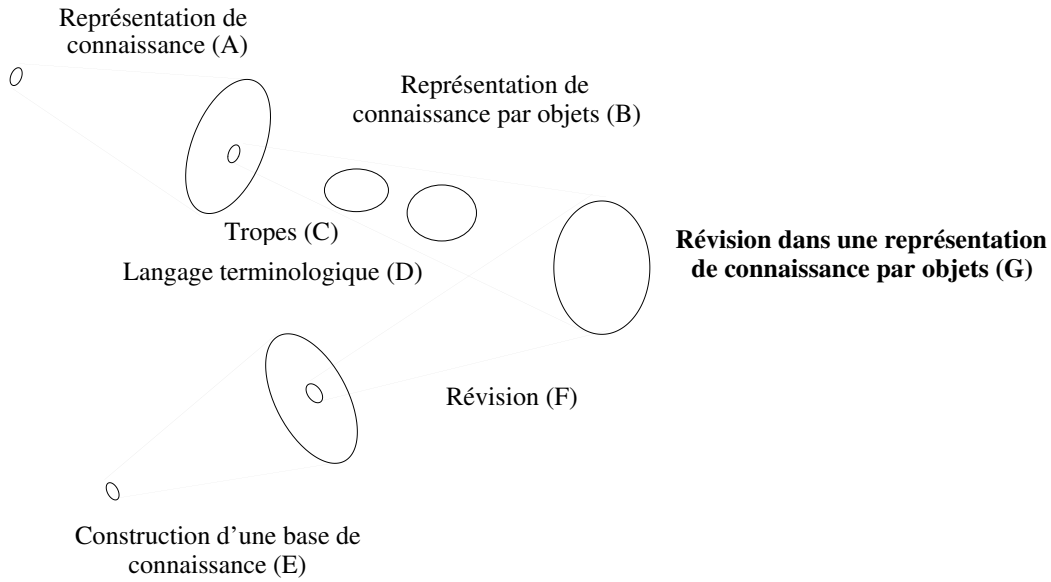


Figure 1 : Organisation du chapitre de l'état de l'art : la représentation de connaissance se précise jusqu'aux représentations de connaissance par objets avec les exemples du système de représentation TROPES et d'un langage terminologique; nous présentons d'autre part les stratégies de construction d'une base de connaissance, et en particulier le mécanisme de révision.

Ce chapitre présente succinctement les principes généraux des représentations de connaissance (section A) et plus particulièrement des représentations de connaissance par objets (section B) en détaillant TROPES (section C), un système de représentation de connaissance par objets particulier sur lequel est réalisée l'implémentation, et un langage terminologique (section D). Dans un second temps, nous présentons les travaux sur la construction d'une base de connaissance qui s'apparentent à la révision (section E), ainsi que les principes et stratégies de révision (section F). Nous pouvons alors caractériser les besoins de révision dans les systèmes de représentation de connaissance par objets (section G).

A. Représentation de connaissance

La représentation de connaissance a pour objectif la modélisation d'un domaine particulier qui puisse être manipulée par un ordinateur. Leur particularité réside dans la séparation de la connaissance et des mécanismes d'exploitation, comme l'inférence de nouvelles connaissances. La représentation de connaissance consiste à modéliser les différents éléments du domaine, les relations qui existent entre eux et leur comportement. Cette modélisation doit être, dans l'idéal, indépendante de tout but particulier, ce qui distingue les systèmes à base de connaissance d'autres systèmes de représentation de connaissance tels que les systèmes à base de règles et les systèmes experts (de première génération tout du moins) qui sont quant à eux élaborés pour résoudre un problème particulier. Les systèmes de représentation de connaissance peuvent être considérées comme des encyclopédies dont les composants sont combinés pour résoudre divers problèmes sur le domaine représenté.

B. Représentation par objets

Nous allons présenter très brièvement la notion d'objet, ainsi que les langages de programmation par objets en nous focalisant sur les différences avec les représentations de connaissance par objets. Une synthèse sur la représentation par objets est donné dans [Masini&89].

1. L'objet et les notions dérivées

Le terme "objet" est utilisé en informatique pour exprimer des notions diverses, mais toujours pour décrire un élément détenant la donnée de son état et l'ensemble de ses propriétés appelées champs ; cet élément définit également lui-même son comportement par le biais de méthodes dans les langages de programmation par objets. La notion de *classe* apparaît dans beaucoup de systèmes basés sur les objets. Une classe regroupe un ensemble d'objets dont la caractérisation de l'état et le comportement sont similaires ; une classe est donc un ensemble de champs et de méthodes commun à tous ses objets. Elle est en fait généralement considérée comme un moule pour la création de ses instances, c'est-à-dire qu'elle permet de définir des objets ayant leurs champs et méthodes spécifiés dans la classe.

L'*héritage* est également un concept fondamental des représentations par objets. Les classes sont généralement liées par une relation d'ordre partiel, appelée *relation de spécialisation*, qui définit pour une classe ses sous-classes et ses super-classes. Le mécanisme d'héritage consiste alors à récupérer pour une classe donnée les champs et les méthodes définis dans ses super-classes.

La notion d'objet a d'abord évolué dans les langages de programmation par objets [Masini&89]. Un programme écrit dans un langage de programmation par objets est un ensemble d'objets qui interagissent par l'envoi de messages. Contrairement aux langages de programmation classiques, les données et les procédures sont regroupées dans les objets. Les classes ont un double rôle : un prédicat d'appartenance et un moule pour la création de leurs instances par la description de leur structure. La classe est donc utilisée d'une part dans un souci d'implémentation et d'autre part pour réaliser une abstraction de données. Sur l'héritage se fonde alors une technique de conception descendante des classes, par affinements successifs.

Un programme est évidemment de la connaissance, mais celle-ci est destinée à être exécutée et non à être manipulée par l'extérieur. Ceci est souligné par le principe de l'encapsulation qui n'autorise pas l'accès direct à un objet et tend à fournir à l'extérieur de l'objet un ensemble de méthodes permettant d'y accéder. Ceci permet de faire des modifications internes à un objet sans que le reste du programme en soit affecté, mais, en contrepartie, limite la visibilité.

Au contraire, des formalismes de représentation tels que les réseaux sémantiques [Quillian68], les langages de schémas ou de '*frames*' [Minski75], les graphes conceptuels [Sowa84] et les logiques terminologiques [Brachman&85, Nebel90] se fondent sur la description d'éléments du monde réel, sans se soucier de leur utilisation future et en laissant à disposition l'ensemble des connaissances. Il en est de même de la représentation de connaissance par objets.

2. Représentation de connaissance par objets[1]

Les systèmes de représentation de connaissance par objets (RCO) reprennent les notions d'objets, de classe et d'héritage. La différence par rapport aux langages de programmation par objets se situe au niveau de la sémantique. Un programme est interprété de manière opérationnelle : quel est son déroulement ? quel est son résultat ? Une base de connaissance, au contraire, s'interprète de manière dénotationnelle : que représente tel objet de la base dans le domaine modélisé ? Une classe est une abstraction d'un ensemble d'entités du domaine.

Les systèmes de représentations de connaissance par objets sont divers, mais l'idée est toujours d'organiser la connaissance autour de la notion d'objet et de fournir des mécanismes d'inférence pour compléter la connaissance disponible. Utiliser les objets permet de concentrer les informations concernant une entité dans un objet et d'exploiter les liens de spécialisation.

[1] Cette partie est largement inspirée de [Euzenat98] dans lequel se trouvent des références aux principaux systèmes de représentation de connaissance par objets.

a) Les objets et les attributs

Les objets sont décrits par une fonction associant une valeur à un attribut, la valeur étant une valeur d'un type prédéfini (entier, réel, chaîne de caractères, etc.) ou un autre objet. Ainsi, il peut exister un objet `mon-appartement` décrit par sa surface, son adresse, son nombre de pièces, son propriétaire, etc. Les attributs permettent d'exprimer les propriétés (la surface par exemple), les relations qui permettent de définir la caractéristique d'un objet comme étant un autre objet (le propriétaire par exemple dont la valeur sera un objet du concept individu) et les liens de composition (une pièce est une partie d'un appartement).

Objet : `mon-appartement`

`surface`	70
`nb pièces`	3
`propriétaire`	Jean Martin
`adresse`	2, rue de Chartreuse 38000 Grenoble
`étage`	4

Pour identifier de manière unique un objet, il est possible de définir certains attributs comme étant les *attributs-clefs*. Par exemple, pour un `individu`, le numéro de sécurité sociale sera l'attribut-clef ; pour un `appartement`, ce sera l'adresse et l'étage.

L'attribut d'un objet peut être non seulement une valeur, mais également :

- une *liste de valeurs* (collection ordonnée de valeurs) ; par exemple : les notes obtenues par un étudiant ;
- ou un *ensemble de valeurs* (collection de valeurs non ordonnée et sans répétition) ; par exemple : les pièces d'un appartement.

Les objets sont regroupés au sein de classes qui définissent les traits communs à ces objets, comme dans les langages de programmation par objets.

b) Les classes

Les classes peuvent être définies soit *en extension* en donnant l'ensemble des objets qui leur appartiennent, soit *en intension* par des restrictions sur les attributs des objets qui leur appartiennent.

Dans le second cas, les valeurs d'attributs admissibles pour les objets rattachés à une classe sont définies syntaxiquement par l'intermédiaire de *facettes* ; la facette `domaine` par exemple indique quelles sont les valeurs autorisées. Les facettes disponibles dans l'un ou l'autre systèmes sont les suivantes :

- *dans* : indique le type de l'attribut, qui doit être un sous-type du type de l'attribut dans les super-classes
- *domaine* : indique la liste des valeurs possibles
- *sauf* : indique les valeurs à rejeter
- *intervalles* : indique l'union d'intervalles de valeurs autorisés (lorsque le type est ordonné totalement)
- *cardinalité* : pour les attributs multi-valués (liste ou ensemble) indique l'intervalle dans lequel peut évoluer le nombre de valeurs

Les facettes permettent également de définir l'*inférence de valeur*, c'est-à-dire de déterminer la valeur des attributs des objets. Ces facettes peuvent être de différents types :

- *valeur par défaut* : cette valeur est donnée à l'attribut si aucune autre n'est disponible et c'est la valeur qui est la plus courante pour l'attribut des objets de la classe décrite ;
- *passage de valeur* : la valeur est celle d'un autre attribut ;
- *filtre* : le filtrage est le mécanisme qui fournit l'ensemble des objets qui satisfont les contraintes attachées au filtre ;
- *attachement procédural* : la valeur est retournée par l'exécution d'une fonction prenant l'objet comme argument.

La classification est un autre mécanisme d'inférence qui permet de déterminer pour un objet les classes auxquelles il peut appartenir vu les contraintes posées sur les attributs des classes. Les inférences ne seront pas plus étudiées, ni utilisées dans la suite de ce mémoire.

D'autre part, les classes sont liées par une relation de spécialisation, qui s'interprète de manière ensembliste : tous les objets appartenant à une classe donnée appartiennent également à chacune de ses super-classes. Ceci implique que, plus la classe est spécifique, plus les domaines des attributs doivent être restreints, car les valeurs acceptables pour l'attribut des objets d'une classe doivent également être acceptables dans les super-classes. Le problème de la multi-spécialisation[2], appelé aussi héritage multiple ou multi-généralisation, apparaît alors : si une classe admet des super-classes incomparables, comment doit se faire l'héritage des restrictions des domaines des attributs ? Ce problème est sujet à de nombreuses discussions (pour une synthèse, voir [Ducournau&95]). Si la multi-spécialisation est interdite, on parle alors de mono-spécialisation.

[2] Noter ici que le terme de multi-spécialisation permet de décrire qu'une classe peut en spécialiser plusieurs autres et non qu'il existe plusieurs classes qui la spécialisent.

Nous allons décrire à présent le système de représentation de connaissance par objets dans lequel est réalisée l'implémentation de notre travail, en montrant ses caractéristiques par rapport au cadre général présenté ci-dessus.

C. TROPES

TROPES [Mariño93] est un système de représentation de connaissance par objets qui définit trois types d'entités principales : concept, classe et instance. Nous précisons la spécificité des ces entités par rapport à la description du paragraphe précédent.

1. Concept, classe et instance

Les éléments du domaine modélisé sont représentés par des instances qui sont partitionnés en concepts : une instance appartient à un et un seul concept. Un concept dénote un ensemble d'éléments qui partagent des propriétés ou des moyens de description, représenté par des attributs. Il peut exister un concept `appartement` ou un concept `individu`. Toute instance d'un concept possède les attributs définis pour le concept et ces attributs ont éventuellement une valeur.

Les classes sont décrites *en intension* et forment des *hiérarchies arborescentes* par la relation de spécialisation : l'hypothèse de *mono-spécialisation* est posée et il existe une classe *racine* qui contient toutes les instances du concept et qui ne pose pas de restrictions sur les valeurs d'attribut. La Figure 2 donne un exemple de hiérarchie.

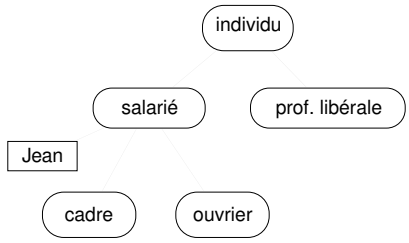


Figure 2 : Exemple de hiérarchie de classes pour le concept `individu` : deux sous-classes sont définies (`salarié` et `profession libérale`), la première admettant à son tour deux sous-classes (`cadre` et `ouvrier`). L'objet `Jean` est instance de la classe `salarié`.

De plus, le fait qu'une instance satisfasse les restrictions posées dans une classe est une condition *nécessaire* à son appartenance à la classe mais *non suffisante*, c'est-à-dire que pour pouvoir être considérée comme un élément de la classe, une instance doit non seulement posséder des valeurs d'attributs qui satisfont les restrictions posées dans la classe mais également y être explicitement rattachée. Les instances d'un concept sont rattachées à au moins une de ses classes, la classe racine par défaut.

Les attributs peuvent être d'un type prédéfini (comme les entiers, les réels, les chaînes de caractères) ou d'un type défini explicitement en indiquant le domaine ainsi que les opérations associées [Capponi95].

2. Les points de vue

Dans une base exprimée en TROPES, une instance peut être vue sous différentes perspectives : un individu peut être considéré selon sa profession, son diplôme, son mode de consommation, etc. ; un appartement peut être considéré selon sa taille, son prix, sa localisation, etc.

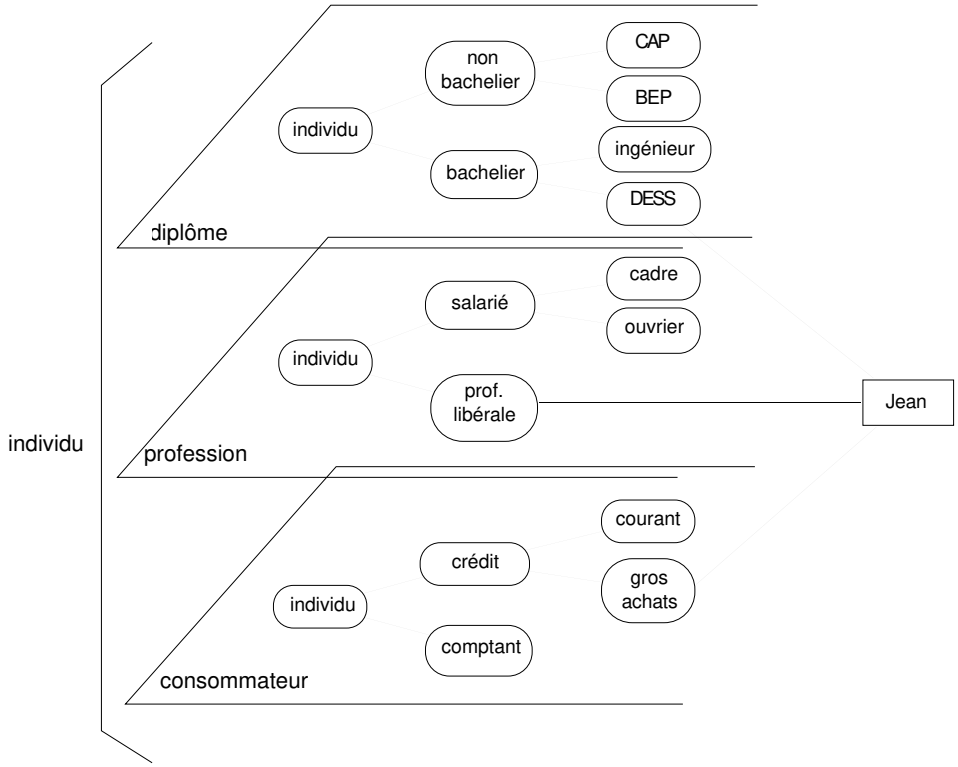


Figure 3 : Exemple de points de vue dans une base TROPES : le concept `individu` est vu sous trois points de vue différents (`diplôme`, `profession` et `type de consommateur`). Une instance particulière du concept (`Jean`) est instance d'une classe de chacun des points de vue.

Une hiérarchie de classes est insuffisante pour représenter cette disparité de perspectives, en particulier à cause de l'hypothèse de mono-spécialisation. C'est pourquoi, dans TROPES, les concepts sont découpés en points de vue qui ne conservent qu'un sous-ensemble des attributs pertinents ; ainsi, le nombre d'années d'ancienneté ne sera pas pertinent dans le point de vue consommateur. Une hiérarchie de classes est définie dans chaque point de vue. Les instances du concept appartiennent à une classe par point de vue (Figure 3).

3. Les passerelles

Les hiérarchies de classes des différents points de vue ne sont cependant pas indépendantes car les instances appartiennent à une classe de chacun des points de vue ; par défaut elles appartiennent à la classe racine de la hiérarchie qui contient toutes les instances du concept. Les *passerelles* permettent de représenter les liens qui peuvent exister entre des classes de différents points de vue en définissant une relation d'inclusion entre l'ensemble d'instances appartenant à un ensemble de classes, appelées *source*, et l'ensemble d'instances appartenant à une classe appelée *destination*. Ainsi, tous les individus appartenant à la classe `cadre` du point de vue `profession` appartiennent également à la classe `bachelier` du point de vue `diplôme` (Figure 4).

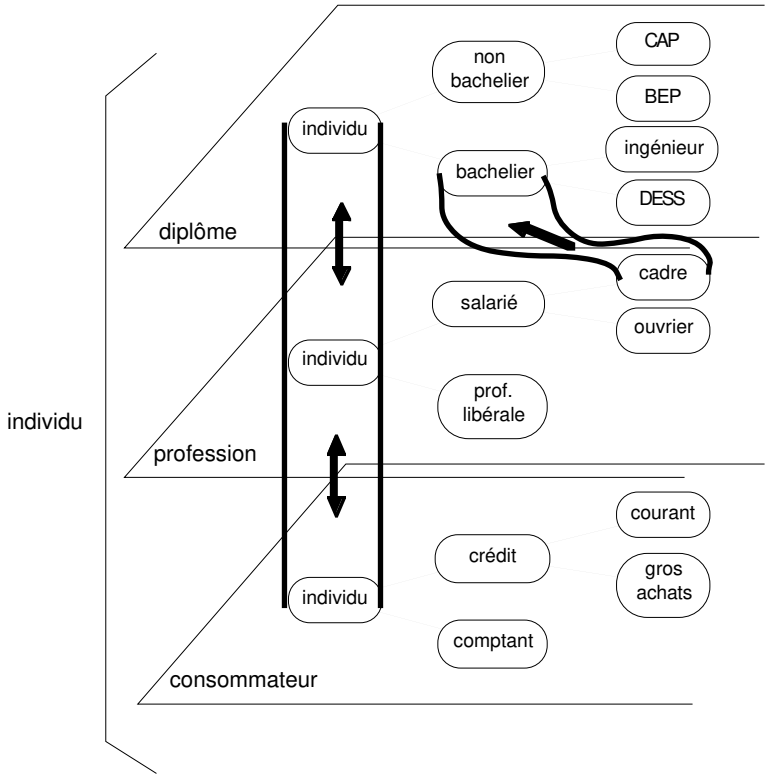


Figure 4 : Exemple de passerelles : les classes racines de chaque concept regroupent les mêmes instances. De plus, tous les cadres sont bacheliers, c'est-à-dire que toutes les instances appartenant à la classe `cadre` doivent aussi appartenir à la classe `bachelier`.

Les passerelles peuvent également être bidirectionnelles, indiquant ainsi que les extensions de deux classes sont identiques.

Le mécanisme de classification, qui consiste rappelons-le à donner pour une instance les classes auxquelles elle peut appartenir, tient compte de l'existence de points de vue et de passerelles. En particulier, lorsqu'une instance appartient à un ensemble de classes qui est

source d'une passerelle, elle appartient aussi à la classe destination de la passerelle, ce qui peut alors impliquer de nouvelles classes d'appartenance pour l'instance.

D. Langage terminologique

Nous allons présenter les grandes lignes du formalisme proposé par [Nebel 90], qui présente un langage terminologique parmi d'autres[3]. Il définit un formalisme terminologique (*TF*) et un formalisme assertionnel (*AF*). Le premier permet de décrire des concepts et des rôles, alors que le second permet de décrire des objets en décrivant leur appartenance à un concept et leur relation par des rôles avec d'autres objets. Nous donnons pour chaque formalisme les grandes lignes de sa syntaxe et de sa sémantique, avant de voir la combinaison des deux.

1. Formalisme terminologique

Les termes regroupent les rôles et les concepts pour lesquels une seule description est donnée. Les rôles peuvent être décrits comme spécialisant un autre rôle ou comme des identificateurs. Quant aux concepts, ils peuvent être :

- primitifs : ils sont définis comme une restriction d'un autre concept ;
- définis : ils sont définis comme une conjonction de concepts, la restriction d'un rôle à un concept particulier ou une restriction sur le nombre d'objet remplissant un rôle.

De plus, il est possible de dire que deux concepts sont disjoints. Les cycles ne sont pas autorisés dans les définitions de termes, car, d'après Nebel, des termes ainsi définis ne peuvent pas se comprendre en fonction de leurs composantes, et il ne s'agit donc pas d'une définition.

La sémantique est basée sur la théorie des modèles, l'idée étant de représenter un ensemble d'objets constituant un domaine. Les concepts sont alors interprétés comme des ensembles d'objets et les rôles comme des relations entre ces objets. De plus, l'interprétation d'un concept primitif est inclue dans l'interprétation du concept qu'il restreint. L'interprétation d'un concept défini est égale à l'ensemble des entités du domaine qui satisfont les contraintes d'appartenance à ce concept. Une interprétation qui satisfait toutes les contraintes liées aux termes d'une terminologie est appelés *structure sémantique* de la terminologie.

[3] Les raisons du choix de ce langage sont multiples : son formalisme est simple, la sémantique est définie d'une manière qui nous paraît intéressante et il a étudié la révision sur son langage.

2. Formalisme assertionel

Les objets sont introduits en indiquant le concept dont ils sont instance. Des relations sont établies entre les objets par le biais des rôles. Il est également possible de poser des contraintes sur le nombre d'objets liés à un objet donné par un rôle.

Il s'agit de définir ensuite ce qu'est un modèle de description d'un monde. Dans une interprétation, un objet appartient à un concept si son interprétation appartient à l'interprétation du concept. Les restrictions sur le nombre d'objets liés à un objet donné par un rôle sont traduites par des restrictions sur des cardinaux. Une interprétation est un *modèle d'une description* de monde si elle satisfait toutes les descriptions des objets et des rôles.

3. Le langage hybride

Le langage hybride consiste à regrouper les deux formalismes présentés précédemment. Cela ne pose pas de problèmes au niveau de la syntaxe, mais au niveau sémantique, il faut que modèle de description de monde et structure sémantique d'une terminologie coïncident sur l'interprétation des concepts primitifs et des rôles.

Après cette présentation des RCO et en particulier de TROPES et d'un langage terminologique, nous entamons la seconde partie du chapitre concernant les travaux qui sont en rapport avec la révision.

E. La construction d'une base

L'ensemble des connaissances disponibles sur un domaine évolue au cours du temps, au fil des expériences, des observations, des formalisations ou des actions. Cette évolution n'est pas forcément monotone, c'est-à-dire que de nouvelles données peuvent contredire des données antérieures. Il est en effet fort peu envisageable qu'une base sur un domaine un tant soit peu complexe puisse être construite d'une seule traite par un seul concepteur ayant une vision suffisamment cohérente du domaine pour ne jamais introduire de connaissances contradictoires. Et si tel était le cas, la question de l'intérêt de la base se poserait ! En effet, une base de connaissance est intéressante pour regrouper et structurer un ensemble de connaissances, et pour faciliter leur exploitation. Si les concepteurs ont une vision globale et cohérente de l'ensemble des connaissances contenues dans la base, la base leur est d'une utilité restreinte : moins d'apprentissage lors de l'effort de modélisation, inférences de connaissances déjà connues. Des contradictions peuvent donc apparaître et ce, dans des

domaines aussi différents que le diagnostic, la planification, la mise à jour de base de données ou la compréhension du langage naturel, comme l'explique [Ginsberg86].

Nous allons examiner ci-dessous les travaux traitant de la construction de base et de la gestion des problèmes de cohérence qui peuvent survenir, en mettant un accent tout particulier sur le mécanisme de révision. Nous commençons par définir brièvement la révision (paragraphe 1), avant d'étudier les travaux qui s'y apparentent (paragraphes 2, 4 et 5). Dans la section suivante (F), nous détaillons les principes sous-jacents à la révision, ainsi que les différentes stratégies employées.

1. La révision en bref

Lorsque de nouvelles connaissances sont disponibles sur le domaine représenté dans une base, il est légitime de vouloir les ajouter à la base. Ce processus est appelé *révision* de la base lorsque le *domaine* est supposé *fixe*. Mais ces connaissances ne sont pas forcément compatibles avec le contenu précédent de la base. Afin d'avoir à tout instant une base de connaissance consistante[4], soit les derniers ajouts sont rejetés, soit le contenu de la base est modifié pour le rendre compatible avec les nouvelles connaissances. L'un des principes de la révision est que les suppressions nécessaires dans une base pour la réviser suite à un ajout inconsistant doivent être minimales, c'est-à-dire, intuitivement, qu'un maximum des connaissances précédentes de la base doit être conservé. Cette minimalité peut cependant s'exprimer de diverses manières, comme nous le verrons (en section F notamment).

Un ensemble important de travaux s'intéressent ou s'apparentent à cette notion de révision, tout en se différenciant, soit sur la méthode de construction des bases, soit sur le traitement des inconsistances, soit sur le type d'évolution du domaine, soit sur l'objectif des bases à construire. La Figure 5 illustre les alternatives à ces quatre critères.

Nous proposons ci-dessous un aperçu des différentes approches, en les triant selon le critère qui nous a semblé le plus pertinent, sachant que bien évidemment chacun de ces travaux peut se voir selon les différents critères. Le paragraphe 2 est consacré aux méthodes de construction et notamment à la combinaison de bases. Le paragraphe 3 étudie notamment l'utilisation de bases qui renferment des inconsistances. Le paragraphe 4 détaille des problèmes quelque peu différents de la construction d'une base, mais qui utilisent des idées similaires : la caractérisation de concepts en apprentissage, l'évolution de schéma dans les

[4] Précisions que nous appelons base consistante, une base qui ne permet pas de déduire de contradictions et donc qui admet au moins un modèle.

bases de données et la construction d'une taxonomie. Enfin, le paragraphe 5 s'intéresse en particulier à la différence entre un domaine statique et un domaine qui évolue.

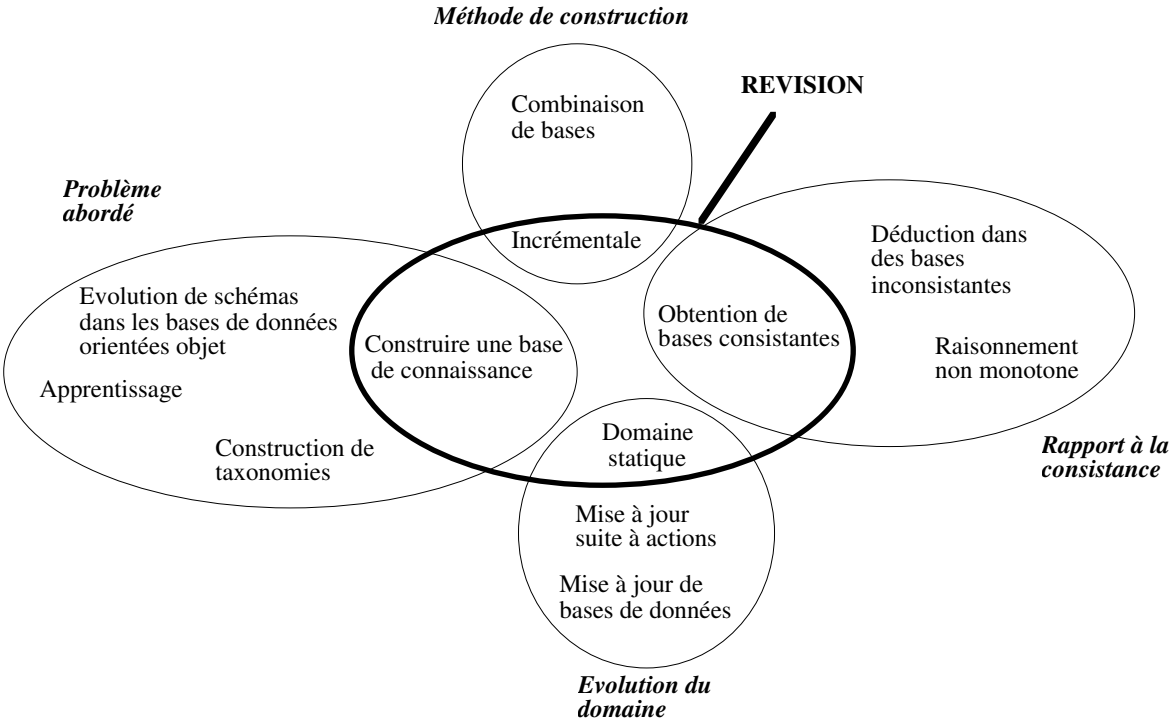


Figure 5 : La construction d'une base et les particularités de la révision. Les différents travaux sont classés selon quatre critères : la méthode employée pour construire la base (paragraphe 2), le rapport à la consistance (paragraphe 3), les problèmes abordés (paragraphe 4) et l'évolution du domaine (paragraphe 5). La révision est caractérisée par rapport à ces différents critères. Évidemment, tous les travaux qui seront présentés ci-dessous selon l'un des critères sont également définissables selon les autres critères, mais le trait le plus caractéristique a été choisi.

2. Méthode de construction

Une base est construite soit en fusionnant des informations provenant de différentes sources, soit de manière incrémentale, par l'ajout successif de connaissances sur le domaine. Le principal problème se pose lorsque l'ensemble des connaissances disponibles n'est pas consistant.

a) Combinaison de bases

Une base de connaissance peut être construite en fusionnant des bases provenant de sources d'informations différentes. La première contrainte est que les représentations employées soient identiques et que dans toutes les bases, une formule dénote la même chose, qui, réciproquement, est toujours exprimée de la même manière. On peut supposer ce problème résolu par ailleurs, même si ce n'est pas forcément trivial. Il s'agit alors de résoudre les inconsistances qui peuvent exister entre les bases. Cette problématique a été abordée par

[Barral&92] qui propose de combiner des théories. Une théorie est une base de connaissances fermée par déduction. De plus, les théories doivent être consistantes avec un ensemble de contraintes d'intégrité. Il utilise la notion de *flock*, c'est-à-dire l'ensemble des sous-ensembles maximalement consistants d'une théorie. Deux stratégies sont envisagées pour obtenir un *flock* qui reflète la combinaison d'un ensemble de théories :

une approche prudente : le *flock* contient les sous-ensembles, de l'union des théories, maximalement consistants avec les contraintes d'intégrité ;

une approche audacieuse : le *flock* contient les sous-ensembles, de l'union des théories, qui sont maximalement consistants et qui sont consistants avec les contraintes d'intégrité

[Barral&92] envisage également la combinaison de théories avec priorités, c'est-à-dire que les théories sont ordonnées selon la validité qui leur est accordée, sachant que les contraintes d'intégrité ont évidemment la priorité maximale. Deux algorithmes sont possibles :

- *bottom-up* : T_1 et T_2 sont combinés en privilégiant T_2, puis combinaison avec T_3 en privilégiant T_3, … puis T_n, enfin combinaison avec les contraintes d'intégrité qui sont privilégiées) ;
- *top-down* : les contraintes d'intégrité sont combinées avec T_n en privilégiant les contraintes, puis avec T_{n-1} en privilégiant le résultat, … puis avec T_1 en privilégiant le résultat précédent.

Le premier algorithme peut conduire à la perte de plus d'informations que le second, mais il représente la voie hiérarchique. [Barral&92] montre aussi que la combinaison de théories permet de prendre en compte l'insertion de formules et la mise à jour.

[Cholvy94] propose de fusionner des sources d'informations selon un principe analogue, en distinguant l'approche suspicieuse – toutes les informations d'une source donnée sont remises en question dès qu'il en existe une qui contredit une source plus fiable – et l'approche confiante – seul l'un des plus petits sous-ensembles d'une source qui contredit une source plus fiable est remis en cause. Cependant, deux sources peuvent ne pas avoir la même crédibilité selon le type d'informations, donc plusieurs ordres sont introduits entre les bases selon le thème : une source peut être plus fiable qu'une autre pour des informations d'un certain thème, mais pas pour un autre.

b) Construction incrémentale

Une base peut aussi être élaborée au fur et à mesure que de nouvelles informations concernant le domaine sont acquises. Tenir compte de ces nouvelles informations dans la base est nécessaire. C'est dans cette optique que nous axons notre travail.

Si les nouvelles informations sont cohérentes avec celles contenues dans la base, peu de problèmes se posent ; dans la plupart des cas, elle seront simplement ajoutées à la base. Nous verrons cependant que si la base est censée représenter un domaine qui évolue, des modifications plus complexes qu'un simple ajout peuvent être effectuées sur la base (voir en particulier le paragraphe 5.b concernant la mise à jour).

Dans certains cas, des informations sont contradictoires avec le contenu précédent de la base, car les concepteurs ne sont pas en mesure d'avoir une vision cohérente du domaine qu'ils représentent, que la connaissance sur le domaine a évolué et certaines hypothèses précédentes sont dépassées ou que le domaine a évolué. Afin de préserver la cohérence de l'ensemble, certaines connaissances doivent être supprimées, ceci est traité par la révision, sauf si le domaine évolue, ce qui est la problématique de la mise à jour.

3. Rapport à la consistance

Il est primordial de pouvoir faire des déductions sensées à partir des bases de connaissance, c'est-à-dire de ne pas pouvoir déduire une chose et son contraire. Pour cela, il faut soit faire en sorte que la base soit un ensemble consistant de connaissance, soit pouvoir faire des déductions même si elle contient des inconsistances. C'est ce deuxième point qui est étudié en détail dans cette section, le premier point correspondant à la révision.

On désignera par *croyances* des connaissances qui représentent ce que l'on croit sur le domaine à un moment donné et qui peuvent donc être remises en cause. Typiquement, les croyances sont les déductions à partir des connaissances sur le domaine.

a) Déduction avec des bases inconsistantes

Des travaux se préoccupent de gérer des bases qui sont inconsistantes. C'est le cas en particulier de Lozinskii [Lozinskii94] qui propose une sémantique pour des systèmes inconsistants. Dans un tel système, la crédibilité d'une formule est évaluée ; elle peut être *valide*, *supportée*, *vague* ou *ambiguë*. Pour cela, il se base sur les sous-ensembles maximalement consistants (noté *mcs*), une formule ayant d'autant plus de chance d'être valide qu'elle l'est dans un plus grand nombre de ces sous-ensembles, sachant que des pondérations peuvent être attribuées aux différents *mcs* selon leur importance. C'est donc un moyen terme entre une approche crédule – toute croyance contenue dans l'un des *mcs* est valide – ou sceptique – seules les croyances contenues dans tous les *mcs* sont valides.

La logique para-consistante consiste à se débarrasser du postulat *ex falso quodlibet* des logiques classiques qui stipule que d'une contradiction tout peut être déduit. Cette approche se fonde sur l'idée que l'inconsistance n'est pas forcément à éviter car des inconsistances sont présentes dans les pensées sans empêcher le raisonnement. Citons les travaux de [Besnard91]

qui définit plusieurs de ces logiques et montre que leur atout par rapport à la révision ou aux logiques non monotones (présentées dans le paragraphe b) est la localité du système de déduction, ce qui permet de ne pas examiner tout le contenu de la base.

b) Raisonnement non monotone

Dans les logiques para-consistantes, le raisonnement est monotone car on ne supprime pas de déductions faites auparavant. Au contraire, le raisonnement non monotone consiste à déduire tout ce qui est possible de l'être et si des inconsistances apparaissent suite à des ajouts dans la base, de supprimer certaines des déductions précédentes. D'après [Makinson&91], le raisonnement non monotone et la révision sont les deux faces d'une même pièce. Les travaux sur le raisonnement non monotone s'apparentant le plus à la révision sont présentés ci-dessous.

[Reiter80] a introduit le raisonnement par défaut afin de tenir compte d'une représentation incomplète du monde. Les conclusions d'une règle de défaut sont considérées vraies tant que le pré-requis qui est associé à la règle est consistant ; dès qu'une inconsistance apparaît, les conclusions sont retirées. Le problème de la révision se pose dès que les croyances produites, les connaissances qui sont déduites de celles qui sont explicitement données, sont inconsistantes. [Boutilier94] a mis en évidence que le raisonnement par défaut peut être vu comme un cas particulier de la révision de croyances, en considérant que les hypothèses faites dans un système de raisonnement par défaut constitue un ensemble de connaissances à réviser si le dernier ajout est inconsistant.

[Brewka89] propose un cadre, étendant celui de Poole [Poole88], permettant de raisonner sur un ensemble inconsistant de formules en considérant ses sous-ensembles consistants *préférés*. Cette notion de préférence est basée sur le découpage de l'ensemble des formules en un nombre fini de classes selon la fiabilité qui leur est accordée par le concepteur de la base, l'ordre entre les ensembles étant lexicographique : un sous-ensemble est alors préféré s'il conserve un sous-ensemble maximalement consistant de formules de la classe des formules les plus fiables, puis le maximum des formules de la seconde classe, etc. Il étudie également un ordre partiel strict sur les formules afin de tenir compte des formules qui ne sont pas comparables. Ces ordres permettent de représenter les priorités qui peuvent exister entre formules et de définir un cadre logique pour le raisonnement par défaut.

Les inférences non monotones sont également reliées à la révision dans [Benferhat&93] qui détermine les sous-bases préférées en faisant intervenir le cardinal des ensembles, afin de réduire le nombre de sous-bases préférées. Ne considérer que les ensembles dont la cardinalité est maximale peut être une idée naturelle de minimalité. [Benferhat&93] définit un ordre lexicographique ($<<_{lex}$) sur les formules qui combine la cardinalité et des priorités sur les

formules. Cet ordre est un pré-ordre complet, c'est-à-dire que l'ensemble S des formules est découpé en une collection $(S_1,S_2,\ldots S_n)$ de sous-ensembles ordonnés strictement selon le degré de priorités des formules qu'ils contiennent ; S_1 (resp. S_n) contient les formules ayant la plus haute (resp. basse) priorité. Formellement, cela se traduit par :

pour tout A et B, $A<<_{lex}B$ ssi il existe i tel que $|A\cap S_i|<|B\cap S_i|$ et pour tout $j<i$ $|A\cap S_j|=|B\cap S_j|$

La relation d'inférence est alors :

$\alpha\vdash_{lex}\beta$ ssi $A\cup\{\alpha\}\vdash\beta$ pour tout A sous-base consistante avec α, maximale par rapport à $<<_{lex}$.

Une telle définition satisfait la propriété de monotonie rationnelle, exprimée par [Gärdenfors&93], et permet d'éviter de perdre trop de formules ; certains systèmes de raisonnement par défaut sont en effet touchés par le *drowning effect* : des formules ne sont plus valides car elles ne sont pas vraies dans *toutes* les sous-bases.

c) Obtention de bases consistantes

Plutôt que de gérer des bases contenant des connaissances inconsistantes, il peut être intéressant de ne traiter que des bases consistantes, ce qui nécessite de définir une opération permettant de supprimer les inconsistances. Cette idée correspond à la volonté de modéliser un domaine particulier : en effet, dans l'ensemble des connaissances, il y en aura forcément certaines qui ne seront pas vérifiées dans le domaine représenté, ou plutôt, qui ne correspondront pas à la vision actuelle du domaine, et donc qui pourront être supprimées afin de préserver la consistance de l'ensemble. C'est l'hypothèse faite lorsque l'on parle de révision ; la section F montre comment ce problème est résolu.

4. Problème abordé

Un certain nombre de travaux s'apparentent à la construction de base de connaissance.

a) Apprentissage

L'un des buts de l'apprentissage est de construire une théorie permettant de caractériser des concepts, afin de pouvoir décider si une entité donnée appartient à un certain concept. Cette construction utilise un ensemble d'exemples positifs – ceux qui doivent être couverts par la caractérisation – et négatifs – ceux qui doivent être rejetés. Ce processus doit être incrémental afin de prendre en compte les nouveaux exemples en modifiant les hypothèses. Un mécanisme de révision est donc nécessaire en particulier lorsqu'un exemple négatif est couvert à tort par la caractérisation et qu'il faut donc la spécialiser. Cette problématique a été étudiée par Wröbel [Wröbel93] qui propose un critère de minimalité et de nouveaux postulats permettant d'éviter

les problèmes tels que la manipulation de théories infinies (car fermées par déduction), la non satisfaction du principe de maintien des justifications et l'obtention de théories complètes.

[Wröbel94] met de plus en évidence la possibilité d'utiliser la révision de théories afin de former de nouveaux concepts : des agrégations d'exemples sont proposées et l'utilisateur décide de garder ou non le concept formé de l'ensemble d'exemples positifs et négatifs.

b) Évolution de schéma dans les bases de données orientées objet

[Barnejee&87] s'intéresse à l'évolution de schéma dans les bases de données orientées objet : changements de la définition des classes ou changement de la hiérarchie de classes. Ces changements doivent respecter un certain nombre d'invariants : le graphe de classes est orienté, acyclique et connexe et admet une racine ; un même nom ne représente qu'une entité de la base, les domaines des classes sont compatibles avec ceux des super-classes, etc. Il détaille un ensemble de règles permettant de préserver au mieux ces invariants lors de modifications de schéma ; il repère également des modifications qui sont inconsistantes mais qui ne sont pas traitées (par exemple donner une valeur par défaut qui n'appartient pas au domaine de la classe). L'implémentation a été effectuée dans le système ORION.

Pour le système GemStone, relativement plus simple que ORION (notamment mettant en œuvre l'héritage simple et non l'héritage multiple), une méthode similaire a été développée [Penney&87]. La différence apparaît principalement lors de la modification de classes : ORION nécessite une vérification des références à un objet à chaque accès à cet objet, alors que dans GemStone, tous les objets sont mis à jour lors de la modification.

[Odberg94] prend en compte des modifications supplémentaires de bases de données orientées objet dans lesquelles, de plus, les objets sont représentés selon plusieurs perspectives. Il traite la compatibilité au niveau de la classification dans les différentes perspectives, du comportement des objets et de l'ensemble des attributs des objets. Il prend en compte les modifications de la base aussi bien suite à des modifications intervenues dans le domaine modélisé qu'à de nouvelles perceptions de ce domaine.

Cependant, dans ces travaux, lors de l'apparition de réelles contradictions – par exemple si une sous-classe ne définit pas des restrictions plus spécifiques que ses super-classes –, aucune solution n'est proposée.

[Theodoratos95a-b] se penche sur la mise à jour de schémas orientés-objet traduits en théories logiques monadiques du premier ordre (logique ne comportant que des prédicats unaires et pas de fonction, ni de constante). Il étudie deux approches syntaxiques nommées SOT (Set Of Theories) et WIDTIO (When In Doubt Throw It Out). La première consiste à rechercher l'ensemble des possibilités de théories mises à jour, alors que la seconde fait

l'intersection de toutes les théories possibles et alors seul ce qui est vrai dans toutes les solutions est gardé. Le problème de la mise à jour d'un schéma est co-NP-complet et il propose des algorithmes incrémentaux, ainsi que des restrictions (sur le nombre de faits, la forme des faits, etc.) qui rendent cette détermination polynomiale.

c) Construction de taxonomies

La construction d'une base peut se traduire par l'accumulation d'un ensemble de connaissances, mais également par la construction de nouveaux concepts lorsque l'on ne considère plus des ensembles de formules mais des taxonomies. En effet, une implication logique peut être considérée comme une caractérisation en remplaçant le 'si ' implicite lié à l'opérateur de conséquence par un 'si et seulement si'. La formule :

mammifère(x) <– donne-du-lait(x) $\wedge$ porte-ses-petits(x)

peut être modifiée en :

mammifère(x) <–> donne-du-lait(x) $\wedge$ porte-ses-petits(x)

Ceci est un exemple de l'hypothèse de monde clos qui conduit à raisonner de manière non monotone, les déductions faites grâce à cette hypothèse peuvent être remises en question. [Foo95] a notamment mis en évidence le besoin d'étendre les taxonomies en raffinant les classes soit en définissant des sous-classes, soit en les décomposant (une classe est remplacée par d'autres). Il s'agit alors d'utiliser les critères de minimalité similaires à ceux donnés en révision pour choisir parmi les alternatives de modification d'ontologie.

5. Évolution du domaine

L'objectif d'une base de connaissance peut être de représenter un domaine statique, ou tout du moins un domaine pour lequel l'évolution n'est pas prise en compte, c'est l'objectif de la révision. Au contraire, on peut vouloir suivre l'évolution du domaine, ce qui est en particulier le cas dans les bases de données ; on parle alors de mise à jour. Le dernier paragraphe met un accent tout particulier sur les différences de traitement d'une nouvelle connaissance dans une optique de révision ou de mise à jour.

a) Mise à jour de bases de données

Les bases de données, comme les bases de connaissance se construisent de manière incrémentale afin de refléter le domaine modélisé. Les modifications des bases visent à rendre compte des modifications qui sont intervenues dans le domaine, la modification de l'ancienneté d'un salarié par exemple.

[Fagin&83] étudie la sémantique de la mise à jour de théories, de bases de données et des vues dans les bases de données, lors de l'ajout ou de la suppression d'un fait. Il opte pour la mise à jour de théories non fermées par déduction afin de privilégier les faits aux formules déduites, c'est-à-dire pour obtenir des mises à jour qui conservent au maximum les faits sans se soucier de ce qui était précédemment déductible. Il introduit une notion de minimalité des modifications grâce à l'ordre suivant :

une théorie T_1 est une mise à jour d'une théorie T qui fait moins de modifications qu'une théorie T_2 si l'obtention de T_1 requiert moins de suppressions que l'obtention de T_2 ou autant de suppressions mais moins d'ajouts.

S'il existe plusieurs théories T_i 1≤i≤n, mises à jour minimales de la théorie initiale T, la théorie résultat T' sera telle que l'ensemble de ses modèles soit l'union des modèles de toutes les théories minimales ; pour des théories finies, le résultat est :

$$T' = \bigvee_{i=1}^{n} T_i = \{\tau_1 \vee \ldots \vee \tau_n \text{ tels que } \tau_i \in T_i, 1 \leq i \leq n\}$$

Ce qui précède doit être étendu afin de traiter la mise à jour de bases de données, car il existe des contraintes d'intégrité qui ne doivent pas être supprimées. Il faut donc établir un ordre de priorité entre tous les faits de la base, la minimalité étant alors étendue afin que la mise à jour conserve le maximum de faits parmi les plus prioritaires. L'administrateur du système est chargé de définir cet ordre dont dépend le résultat des mises à jour. Cependant, le mécanisme proposé ne peut être implémenté car l'insertion est NP-difficile et les suppressions provoquent une explosion combinatoire de disjonctions.

De nombreux travaux étudient la mise à jour de bases de données relationnelles ou déductives ([Abiteboul88] fournit un panorama des travaux sur la mise à jour de vues dans les bases de données).

La contraction de bases de données, c'est-à-dire le retrait de données a été également étudiée [par exemple, Aravindan&94, Kakas&90]. Ils se rapprochent de la révision si celle-ci est exprimée à l'aide d'un opérateur de contraction (lorsqu'elle est décomposée en une étape de contraction de la base en une base consistante avec la connaissance à ajouter, suivie de l'ajout de cette dernière connaissance). La révision se traduit alors par la suppression de l'opposé de la formule à ajouter. [Aravindan&94] tente de concilier la rationalité sous-jacente à la révision et au raisonnement non monotone, et l'efficacité de l'implémentation de la mise à jour de schémas. La base est contractée en supprimant les formules composant les preuves minimales de la formule à supprimer, c'est-à-dire les formules qu'il suffit de retirer afin de couper la chaîne de déduction.

b) Mise à jour suite à des actions

[Winslett 88] propose un opérateur *pma* (pour Possible Models Approach) qui se fonde sur les modèles. Il s'intéresse aux modifications suite à des actions sur le monde, dans un langage de logique du premier ordre dans lequel les bases peuvent contenir un ensemble de formules qui ne peuvent être supprimées. Une action se traduit par l'ajout d'une nouvelle formule dans la base ; ainsi, dans une base représentant l'état d'un monde de blocs {*a* est sur le sol, *a* est sur *b* et *b* est sur *c*}, une action peut être "déplacer *b* sur le sol" ; mettre à jour la base revient à la modifier pour prendre en compte cette action. L'ensemble d'états possibles du monde après une action donnée est l'union de tous les modèles compatibles avec le nouvel ensemble de formules et qui diffèrent minimalement d'un modèle de la théorie initiale, c'est-à-dire dont le moins d'atomes – au sens de l'inclusion ensembliste – sont modifiés. Le problème est qu'il n'existe pas d'algorithme pour *pma* dans le cas général, car il n'est pas possible de calculer tous les modèles compatibles avec un ensemble de formules.

[Cordier&94] propose un modèle formel pour la transition d'état cognitif (ensemble de faits sur un monde en évolution) lorsque le domaine évolue ; la révision consiste alors à effectuer toutes les transitions qui sont possibles, puis à faire l'intersection des états possibles.

c) Différence mise à jour / révision

Nous avons abordé les problèmes de mise à jour de bases de données (paragraphe a) et suite à des actions (paragraphe b). Donnons une intuition de la différence entre la révision et la mise à jour. La différence cruciale concerne l'objectif fixé : alors que la mise à jour consiste à rendre compte d'une modification qui est intervenue dans le monde, soit par une action délibérée, soit par une évolution naturelle du domaine, la révision s'attache à tenir compte d'une nouvelle connaissance sur le domaine qui est (considéré comme) immuable. C'est pour cela que dans les bases de données, le problème est la mise à jour. Ces deux problèmes ont évidemment en commun la recherche des informations précédentes à retrancher afin de tenir compte des nouvelles données, mais les principes ne pourront être identiques.

[Katsuno&91b] montre que les postulats donnés pour la révision par [Alchourrón&85] (énumérés au paragraphe F.1.a) ne correspondaient pas à la mise à jour et en donne de nouveaux conformes à ce qui est attendu d'un opérateur de mise à jour. La différence se situe dans le traitement d'une donnée consistante avec la base actuelle. Lorsqu'une connaissance ne contredit pas le contenu précédent de la base, la révision de la base consiste uniquement à l'ajouter. Par contre, lors d'une mise à jour, ce n'est pas le cas, car des modifications supplémentaires sont effectuées. Considérons l'exemple suivant tiré de [Katsuno&91a] (Figure 6) : soient deux faits *a* et *b* ; *a* dénote le fait qu'un livre est posé sur le sol et *b* qu'un

magazine est posé sur le sol. Soit la base $(a \wedge \neg b) \vee (\neg a \wedge b)$ dénotant qu'il y a sur le sol ou le magazine ou le livre de manière exclusive.

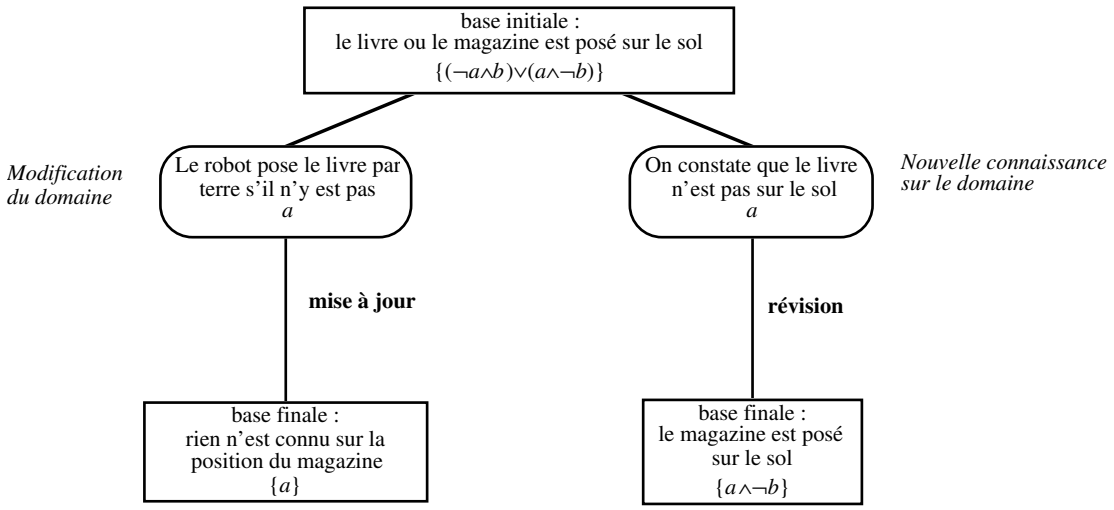


Figure 6 : selon qu'une formule reflète une action effectuée sur le monde ou une nouvelle information, le résultat escompté n'est pas le même et donc deux opérations distinctes sont nécessaires : dans le premier cas, il s'agit de mise à jour, dans le second de révision.

Si on commande à un robot de faire en sorte que le livre soit par terre, a est à présent vrai. L'opérateur de mise à jour *pma*, décrit par Winslett [Winslett88], qui se fonde sur la modification de modèles, permet d'obtenir la base (a), c'est-à-dire que rien ne peut être déduit sur l'emplacement du magazine, ce qui semble raisonnable. Au contraire, la révision de la base se résume à ajouter a et on obtient la base $(a \wedge \neg b)$. Il est donc possible de déduire que le magazine n'est plus sur le sol, c'est-à-dire que s'il s'y trouvait, il aurait dû se déplacer suite au déplacement du livre ! La révision de la base reflète, non pas l'état du monde après une action, mais ce qui est connu du monde suite à une nouvelle donnée ; ainsi dans l'exemple, si on sait que ou le magazine ou le livre est sur le sol et que l'on voit que le livre est sur le sol, de la base révisée on va pouvoir déduire que le magazine n'est pas sur le sol. Ceci est un exemple simple montrant que la révision et la mise à jour diffèrent sur ce que dénote une nouvelle formule et donc que les moyens employés doivent être différents : la révision ajoute la nouvelle formule alors que la mise à jour modifie les valeurs de vérité des atomes.

Ainsi, la modification d'une base qui représente un monde sur lequel des actions sont effectuées ou un monde statique ne requiert pas la satisfaction des mêmes postulats, en particulier dans le cas de l'ajout d'une connaissance consistante avec la base existante.

F. Comment réviser une base ?

Après ce tour d'horizon de travaux sur la construction de bases, en rapport avec la révision, il s'agit d'étudier celle-ci dans le détail ; des synthèses sur les travaux de révision sont également données dans [Sombé94] et [Papini96], et pour la logique classique dans [Lévy94]. Rappelons que la révision consiste à préserver la consistance d'une base de connaissance qui représente un domaine statique et qui est construite de manière incrémentale. De plus, tous les travaux sur la révision se préoccupent de bases exprimées dans un formalisme logique.

Lors de la révision, la modification de la base initiale doit suivre des principes énoncés dans la littérature dont certains sont sujets à controverse (paragraphe 1.a). Une synthèse regroupant les différents principes est donnée (paragraphe 1.b). L'énoncé de ces principes n'est cependant pas suffisant pour caractériser de manière unique la révision. De nombreuses stratégies de révision ont été explorées (paragraphe 2). Une des idées intuitives concernant les propriétés de la révision est que l'on souhaite perdre le moins de connaissances possibles. Cette notion de minimalité, si elle est communément admise, est également interprétée de multiples manières, comme nous le verrons tout au long du détail des différentes stratégies de révision.

1. Principes de révision

Nous allons nous attacher à décrire les principes énoncés sur la révision ; ces principes visent à caractériser les intuitions sous-jacentes à la notion de révision. Différents auteurs sont étudiés avant de donner un récapitulatif des principaux postulats communément admis et de ceux qui sont contradictoires entre eux.

a) Principes de révision au fil de la littérature

Alchourrón, Gärdenfors et Makinson [Alchourrón&85] furent les premiers à se pencher sur la définition formelle d'opérateurs de révision qui s'appliquent sur des théories[5], c'est-à-dire des ensembles de propositions fermés par déduction. $\neg$ représente la négation, Cn la clôture déductive d'un ensemble de formules et $\dot{+}$ un opérateur de révision. Les postulats pour la révision sont les suivants (ils seront appelés AGM dans la suite) :

Soient A une théorie et x, y deux propositions consistantes.

(G1) $A \dot{+} x = \mathrm{Cn}(A \dot{+} x)$

(G2) $x \in A \dot{+} x$

[5] A noter que le terme *théorie* est utilisé ici pour dénoter un ensemble fermé par déduction, ce qui n'est pas toujours le cas, en particulier dans [Fagin&83].

(G3) $A \dot{+} x \subseteq \mathrm{Cn}(A \cup \{x\})$

(G4) si $\neg x \notin A$, alors $A \cup \{x\} \subseteq A \dot{+} x$

(G5) si $\neg x \notin \mathrm{Cn}(\emptyset)$, alors $A \dot{+} x$ est consistant

(G6) si $\mathrm{Cn}(x)=\mathrm{Cn}(y)$, alors $A \dot{+} x = A \dot{+} y$.

(G7) $A \dot{+} (x \wedge y) \subseteq (A \dot{+} x) \cup \{y\}$

(G8) si $\neg y \notin A \dot{+} x$, alors $(A \dot{+} x) \cup \{y\} \subseteq A \dot{+} (x \wedge y)$

(G1) exprime que le résultat de la révision doit toujours être fermé par déduction. (G2) traduit le fait que la dernière proposition doit appartenir à la théorie révisée, c'est-à-dire que l'on accorde toujours le plus d'importance à la dernière proposition ajoutée à la base. (G3) exprime que lors de la révision, on n'ajoute pas plus de proposition qu'en faisant l'union ; dans le cas où la proposition est consistante avec la base, (G4) combiné à (G3) signifie que la révision revient exactement à faire l'ajout. (G5) stipule que dès que la proposition à ajouter n'est pas inconsistante en elle-même, la base révisée doit être consistante. (G6) exprime que la forme syntaxique de la proposition ajoutée ne doit pas influer sur le résultat de la révision. (G7) et (G8) expriment la minimalité de la révision.

Katsuno et Mendelzon [Katsuno&91a] ont établi un ensemble de postulats équivalents s'appliquant sur des bases de connaissance non fermées par déduction et s'exprimant sémantiquement. On dira qu'une base est *satisfaisable* si elle admet au moins un modèle et qu'elle *implique* une proposition si cette proposition est vraie dans tous les modèles de la base. Soit $\circ$ l'opérateur de révision, K une base de connaissance et x une proposition :

(K1) $K \circ x$ implique x

(K2) si $K \wedge x$ est satisfaisable, alors $K \circ x = K \wedge x$

(K3) si x est satisfaisable, alors $K \circ x$ est également satisfaisable

(K4) si $x \equiv y$, et $K \equiv K'$ alors $K \circ x = K' \circ y$.

(K5) $(K \circ x) \wedge y$ implique $K \circ (x \wedge y)$

(K6) si $K \circ (x \wedge y)$ est satisfaisable, alors $K \circ (x \wedge y)$ implique $(K \circ x) \wedge y$

(K1-4) sont équivalents à (G1-6) et (K5) et (K6) sont respectivement équivalents à (G7) et (G8). Ces deux derniers postulats sont liés à la condition stipulant que la révision doit se faire de manière minimale.

[Nebel90, 7.1.2] donne également une liste de principes informels pour la révision dans une base de connaissance, et plus précisément pour la définition de deux opérations TELL et FORGET qui, respectivement, ajoute et supprime une connaissance dans la base :

(N1) Adéquation du langage de requête : la connaissance à supprimer ou à ajouter doit être interprétable dans le langage de la base.

(N2) Indépendance de la syntaxe : si l'expression de deux connaissances est équivalente, les opérations auront la même base pour résultat.

(N3) Clôture : le résultat des opérations est une base de connaissance unique, représentable dans le formalisme employé.

(N4) Succès : les opérations doivent réussir ; après TELL la nouvelle connaissance doit être dérivable de la base et après FORGET la connaissance ne doit plus pouvoir être déduite

(N5) Changement minimal : la modification de la base de connaissance doit être minimale, selon une mesure de minimalité à définir.

(N6) Efficacité : les opérations de révision doivent être aussi efficaces que possibles et en particulier leur complexité ne doit pas dépasser celle des algorithmes d'inférence.

Son principal ajout par rapport à AGM est le principe d'efficacité. De plus, il fait intervenir une mesure de minimalité. Il prône l'indépendance de la syntaxe dans [Nebel90, 7.1.2], et [Nebel92] étudie une approche syntaxique de la révision, où il met en évidence que la syntaxe permet de définir un ordre sur les connaissances qui peut être significatif si la syntaxe de la base reflète l'ordre d'importance des connaissances.

[Dalal88] a également énoncé des principes permettant de caractériser l'intuition que l'on peut avoir de la révision dans une base de connaissance. Il prône en particulier l'indépendance de la révision par rapport à la syntaxe de la base.

(D1) Adéquation de la représentation : la base révisée doit être représentée dans le même formalisme que la base initiale afin de pouvoir lui appliquer les mêmes opérateurs.

(D2) Non pertinence de la syntaxe : la base révisée ne doit pas dépendre ni de la syntaxe des anciennes connaissances, ni de celle des nouvelles. En particulier, il n'y a pas de distinction entre un ensemble de formules (avec une conjonction implicite) et la conjonction de ces formules, puisqu'ils sont équivalents logiquement.

(D3) Préservation de la consistance : si la base initiale et la nouvelle formule sont toutes deux consistantes alors il en est de même de la base révisée.

(D4) Primauté de la nouvelle information : la base révisée doit permettre de déduire la nouvelle information ajoutée, ce qui implique que c'est celle qui est considérée comme la plus crédible.

(D5) Préservation des anciennes connaissances : le maximum de connaissances de la base initiale doit être gardé dans la base révisée. Il s'agit donc de réviser la base par une

modification minimale de la base initiale. Comme cela a déjà été signalé, cette notion de minimalité admet différentes interprétations.

(D6) Équité : si plusieurs bases révisées satisfaisant les principes énoncés ci-dessus sont possibles, il faut que le choix entre elles soit explicite et reproductible, afin d'éviter le non déterminisme.

Afin d'ordonner les postulats qui existent dans les différents travaux traitant de révision, de mise à jour ou de raisonnement non monotone, [Hansson96] étudie quatre critères :

(H-S) succès : la dernière donnée doit appartenir à la base,

(H-C) consistance : la base résultat doit être consistante,

(H-P) préservation : la perte de données non nécessaire doit être évitée,

(H-R) confinement *:* l'ajout de données superflues doit être évité.

Pour chacun, il donne différentes versions classées de la moins contraignante à la plus contraignante. Soit K une base fermée par déduction et $K*x$ la base résultant de la révision de K par x.

Succès :

(H-S1) si $x \in K$ et $\neg x \notin K$ alors $x \in K*x$.
la proposition appartient à la base si elle est déjà déductible.

(H-S2) si $\neg x \notin K$ alors $x \in K*x$.
la proposition appartient à la base si sa négation n'était pas déductible.

(H-S3) si $\neg x \notin \mathrm{Cn}(\emptyset)$ alors $x \in K*x$.
la proposition appartient à la base si elle n'est pas inconsistante en elle-même.

(H-S4) $x \in K*x$.
la proposition appartient à la base dans tous les cas.

Consistance :

(H-C1) si $x \in K$ et $\neg x \notin K$ alors $K*x$ est consistant.
la base révisée est consistante si la proposition est déjà déductible.

(H-C2) si $\neg x \notin K$ alors $K*x$ est consistant.
la base révisée est consistante si le contraire de la proposition n'était pas déductible.

(H-C3) si $\neg x \notin \mathrm{Cn}(\emptyset)$ alors $K*x$ est consistant.
la base révisée est consistante si la proposition n'est pas inconsistante en elle-même.

(H-C4) $K*x$ est consistant.
la base révisée est consistante dans tous les cas.

Préservation :

(H-P1) si $x \in K$ et $\neg x \notin K$ alors $K \subseteq K*x$.
la base révisée contient la base initiale si la proposition est déjà déductible.

(H-P2) si $\neg x \notin K$ alors $K \subseteq K*x$.
la base révisée contient la base initiale si le contraire de la proposition n'était pas déductible.

(H-P3) si $y \in K \setminus (K*x)$ alors $\neg x \in K$ et $\neg y \in K*x$.
les seules propositions supprimées sont celles dont on peut déduire le contraire de la base révisée.

(H-P4) $K \subseteq K*x$.
la base révisée contient la base initiale dans tous les cas.

Confinement *:*

(H-R1) si $x \in K$ et $\neg x \notin K$ alors $K*x \subseteq K$.
la base révisée est inclue dans la base si la proposition est déjà déductible.

(H-R2) $K*x \subseteq \mathrm{Cn}(K \cup \{x\})$.
la base révisée est inclue dans la clôture déductive de l'union de la base et de la proposition ; dans le cadre logique cela ne pose aucune contrainte s'il y a inconsistance.

Hansson a établi les ensembles maximaux de contraintes que peut satisfaire un opérateur de révision[6], soit (H-S4, H-C3, H-P3, H-R2), (H-S3, H-C4, H-P3, H-R2) et (H-S4, H-C2, H-P4, H-R2). En effet, il est par exemple impossible qu'un opérateur de révision satisfasse à la fois H-S4 et H-C4, car si la connaissance à ajouter est inconsistante l'un des deux est nécessairement faux.

b) Synthèse des différents principes

Suite au détail des principes, cette synthèse vise à mettre en évidence les caractéristiques attendues de la révision, ainsi que les divergences majeures. Les postulats décrits dans le paragraphe précédent seront indiqués entre parenthèses pour chacun des principes.

Formalisme de représentation (G1-D1-N1-N3) : suite à la révision, une base doit être exprimée dans le même formalisme que la base initiale, aussi bien le type de connaissances (des formules propositionnelles, en logique du premier ordre), que le type de l'ensemble (fermé ou non par déduction par exemple). Nebel met l'accent sur la représentation de la connaissance ajoutée. Ce postulat est unanimement admis et même suffisamment naturel pour ne pas être toujours exprimé.

[6]A noter qu'il parle de mise à jour de bases de données, mais que ces critères correspondent à de la révision (cf le paragraphe E.5.c page 19).

Consistance de la base révisée (G5, K3, D3, H-C) : comme cela a déjà été souligné précédemment, la consistance de la base est la condition *sine qua non* pour pouvoir exploiter les connaissances qui s'y trouvent, exception faite des systèmes qui gèrent les inconsistances (cf. paragraphe E.4.a). Après révision, il faut donc pouvoir garantir la consistance de la base révisée, ou au moins définir les conditions dans lesquelles elle l'est. Généralement, quand la base et la dernière connaissance sont toutes deux consistantes, la base révisée l'est aussi. (H-C3), (G5), (K3) et (D3) sont équivalents.

Ajout de la dernière connaissance (G2, K1, D4, H-S, N4) : l'objectif même de la révision est de tenir compte de la dernière connaissance ; il est donc communément admis que cette connaissance doit être présente dans la base après révision. Mais, que faire en présence d'une connaissance inconsistante (par exemple, $x \wedge \neg x$) ? Cette question mérite une réponse dans tout descriptif d'un opérateur de révision. (G2) et (K1) sont équivalents à (H-S3) car ils ne considèrent la révision que par une connaissance qui est consistante en elle-même. Il n'y a pas révision de la base par une connaissance inconsistante en elle-même.

Les deux principes suivants sont antagonistes :

Préservation du maximum de connaissances (G4, D5, H-P, N5) : ce principe qui exprime l'idée naturelle selon laquelle on souhaite perdre le moins possible de connaissances de la base soulève néanmoins des controverses. Le moins possible de connaissances explicitement introduites dans la base ou appartenant à la clôture déductive ? Les connaissances ont-elles toutes la même importance ou faut-il en privilégier certaines ?

Maintien des justifications : ce principe stipule que toutes les connaissances déduites d'une connaissance qui est supprimée doivent également être supprimées. Prenons l'exemple de base suivant : {"S'il pleut, Jean prend son parapluie", "Il pleut"} ; on peut en déduire que "Jean prend son parapluie". Or, en regardant par la fenêtre, on se rend compte qu'il ne pleut pas. La base révisée par ¬"il pleut" ne devrait plus permettre de déduire "Jean prend son parapluie". Il y a effectivement contradiction avec le principe de préservation du maximum de connaissances, car selon ce principe "Jean prend son parapluie" devrait toujours appartenir à la base.

L'une des différences avec le principe précédent est donc que la révision dans ce cas s'effectue sur des bases (c'est-à-dire contenant les faits et non fermées par déduction), alors que lorsque l'on souhaite perdre le moins de connaissances de la clôture déductives, la révision s'effectue sur des théories.

Pertinence ou non de la syntaxe (G6, K4, D2, N2) : la syntaxe peut être vue comme contenant de l'information ou au contraire comme un obstacle dont il faut s'affranchir

lors de la révision. Ainsi, deux bases équivalentes d'un point de vue logique peuvent, selon le contexte, dénoter exactement la même chose ou exprimer des représentations différentes du monde. Dans le second cas, les bases seront donc révisées de manière différente.

Confinement (G3, K2, H-R) : le maximum de connaissances doit être préservé mais en ajouter de nouvelles n'est pas non plus souhaitable. La base révisée ne doit pas permettre de déduire plus de connaissances que l'union de la base et de la dernière connaissance. Dans le cadre logique, cependant, le postulat (G3) par exemple n'est pas très contraignant lorsque la proposition est inconsistante avec la base car l'union est équivalente à faux et permet donc de tout déduire !

Équité (D6) : plusieurs bases révisées sont généralement possibles et le choix entre elles ne peut être arbitraire, ne serait ce que pour éviter le non déterminisme de l'opérateur de révision. Ce principe peut être lié à celui de préservation, car l'une des possibilités est de réduire l'indéterminisme en introduisant une relation d'ordre entre les connaissances de la base et à choisir la base révisée qui en préserve le maximum parmi les plus importantes.

Efficacité (N6) : Nebel est le seul à se préoccuper de problème de complexité, même si la conception d'un algorithme est également envisagé par Dalal, par la description syntaxique de la révision.

Ces différents principes vont permettre de guider la définition d'un opération de révision ; nous les comparons avec les besoins dans les bases de connaissance à objets dans le paragraphe G.2.a et nous voyons dans quelle mesure ils sont satisfaits pour la révision que nous définissons dans la section III.D.

Voyons à présent comment les opérateurs de révision peuvent être définis.

2. Stratégies de révision

Nous allons étudier des opérateurs de révision définis dans la littérature et nous voyons en particulier comment ils se situent par rapport aux postulats (G1-G8) de [Alchouròn&85]. Chronologiquement, les premiers opérateurs de révision se sont appliqués aux théories, c'est-à-dire aux bases fermées par déduction (paragraphe a). La taille infinie de ces bases posant problème pour la mise en œuvre, la révision de bases finies a été envisagée (paragraphe b). D'autre part, afin de ne pas tenir compte de la syntaxe d'une base pour la réviser, des opérateurs de révision se basent sur les modèles (paragraphe c). Mais ce choix entre sémantique et syntaxe n'est pas toujours fait (paragraphe d). Enfin, des systèmes permettent de caractériser logiquement des opérations de modifications de connaissances en introduisant la notion d'état épistémique (paragraphe e).

a) Modification de théories en logique

[Alchourrón&85] ont en fait défini la révision en fonction d'une opération qu'ils qualifient de plus élémentaire : la *contraction* (notée $\dot{-}$), opération qui consiste à supprimer de la base toutes les formules qui permettent de déduire le contraire de la proposition à ajouter. En reprenant les notations détaillées ci-dessus, l'identité de Lévi caractérise la révision par :

$A \dot{+} x = \mathrm{Cn}((A \dot{-} \neg x) \cup \{x\})$.

Comment cet opérateur de contraction peut-il être défini ?

On note $A \perp x = \{B$, sous-ensemble maximal de A tels que $B \nvdash x\}$

La première possibilité – la contraction maxi-choix (*maxichoice contraction*) – définit $A \dot{-} x$ comme étant un élément arbitraire de $A \perp x$. L'opérateur de révision maxi-choix défini par [Alchourrón&85] satisfait l'un des ensembles maximaux de contraintes donnés par Hansson (H-S4, H-C3, H-P3, H-R2) (paragraphe 1.a). Afin d'éviter ce choix arbitraire, il est tentant de définir la contraction comme étant l'intersection de tous les éléments de $A \perp x$, cette contraction est appelée contraction par intersection complète (*full meet contraction*). Or, cette intersection est en général bien trop restreinte car elle conduit à un opérateur de révision qui ne correspond pas à ce qui est attendu. En effet, si x est inconsistant dans A, on obtient $A \dot{+} x = \mathrm{Cn}(x)$, c'est-à-dire que seules les conséquences de x sont conservées, ce qui semble bien réducteur. L'opération de révision définie ainsi satisfait tout de même les postulats (G1-8). Ces postulats ne sont donc pas suffisants pour garantir que la modification de la théorie est faite de manière satisfaisante.

Entre les possibilités de contraction maxi-choix ou de contraction par intersection complète – c'est-à-dire entre le choix de l'un des sous-ensembles maximaux ou l'intersection de tous ceux qui sont possibles – la contraction peut être définie en se servant d'une fonction de choix, notée χ, qui sélectionne certains éléments de $A \perp x$, et en fait l'intersection. Elle est appelée contraction par intersection partielle (*partial meet contraction*) : $A \dot{-} x = \cap \chi(A \perp x)$. Ces trois postulats sont explicités dans la Figure 7.

Le point délicat est bien sûr de définir χ, ce qui nécessite la définition d'une relation d'ordre entre les sous-ensembles afin de sélectionner les "meilleurs". Pour cela, Gärdenfors et Makinson [Gärdenfors&88] ont introduit la notion de conséquence épistémique (*epistemic entrenchment*) qui définit un ordre $\leq_{EE}$ entre les propositions afin que lors de la contraction les propositions ayant la plus forte valeur épistémique soient les plus difficiles à supprimer. Cette conséquence épistémique ne peut être définie de manière complètement arbitraire. En particulier : si $a \vdash b$, alors $a \leq_{EE} b$, c'est-à-dire que la conséquence d'une proposition est épistémiquement au moins aussi importante que la proposition elle-même.

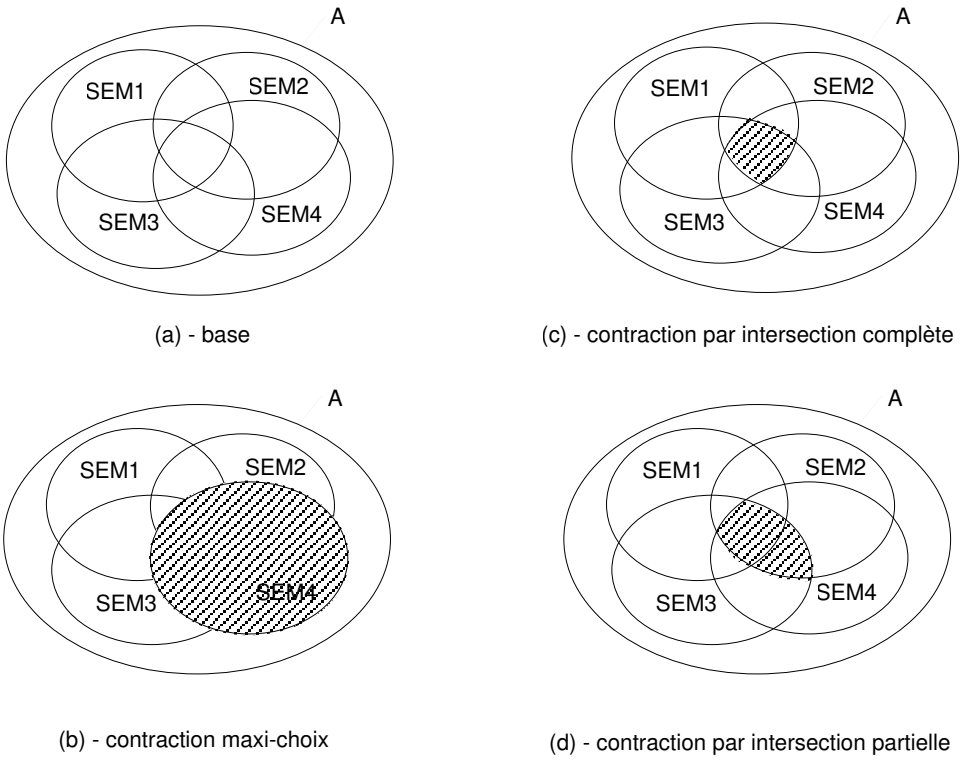


Figure 7 : Considérons une base ayant quatre sous-ensembles maximaux consistant avec une proposition (a). La contraction maxi-choix choisit l'un des sous-ensembles (b). La contraction par intersection complète fait l'intersection de tous les sous-ensembles (c). La contraction par intersection partielle fait l'intersection de certains des sous-ensembles (SEM2, SEM3 et SEM4 - d).

Donc, dans l'exemple, afin de minimiser les pertes, *a* sera supprimé préférentiellement à *b* (car la suppression de *b* nécessite celle de *a*). Notons que ceci va à l'encontre du principe de maintien de justification (paragraphe 1.b), qui privilégierait la suppression de *b* et donc de la proposition *a* qui permet de le déduire. Le problème de cette approche est qu'il n'est pas possible de manipuler en pratique des ensembles fermés par déduction car leur taille est trop importante, voire infinie, et il n'est donc pas possible de mettre en œuvre pratiquement la révision de théories. C'est pourquoi les modifications de bases non fermées par déduction, et finies, sont envisagées, d'autant plus que ceci permet alors de tenir compte du maintien de justification.

b) Révision sur des bases finies

Afin de tenir compte de la validité différente accordée aux propositions (entre des faits et des règles d'intégrité par exemple), [Nebel92] introduit un pré-ordre complet entre les propositions selon la pertinence épistémique (*epistemic relevance*). La base *C* est alors décomposée selon les classes d'équivalence associées à cet ordre. La contraction de *C* par une formule ϕ, notée $C \Downarrow \phi$, consiste alors à préserver le plus grand sous-ensemble de la classe d'équivalence ayant la plus grande importance ne permettant pas de déduire ϕ, puis le

maximum de formules de la classe ayant l'importance suivante, etc. La révision sur des bases avec priorités (*prioritized base revision*) se définit alors pour les bases finies par :

$C \oplus \phi = \mathrm{Cn}((\vee(C\Downarrow\neg\phi))\wedge\phi)$.

Le résultat intéressant est que la classe des opérateurs de révision sur des bases avec priorités est équivalente à la classe des opérateurs de révision sur des théories générées par la pertinence épistémique. Ceci permet d'envisager l'utilisation pratique de la pertinence épistémique puisque les bases ne sont plus nécessairement représentées par des ensembles de formules fermés par déduction, à la condition d'utiliser des méthodes polynomiales pour vérifier la satisfiabilité.

[Weydert92] propose d'étendre la révision sur des bases avec priorités proposée par Nebel en utilisant deux pré-ordres différents pour la dépendance et la préférence entre les formules. Ces ordres permettent de déterminer les sous-ensembles de propositions dont l'intersection est la base révisée. Cet opérateur de révision ne satisfait pas tous les postulats AGM (seuls G1, G3, G4 et G6 sont satisfaits dans le cas général), mais il permet d'exprimer plus facilement la préférence entre les formules.

c) Révision du point de vue des modèles

Selon Dalal [Dalal88], afin d'éviter que la syntaxe d'une base interfère sur sa révision, il faut définir la révision sur les modèles de la base. La base est représentée par un ensemble de formules (éventuellement réduit à un élément) construites sur un ensemble d'atomes. Une interprétation est une fonction qui donne à chaque atome une valeur de vérité. Un modèle est une interprétation dans laquelle toutes les formules sont évaluées à vrai. Afin de réviser une base lorsqu'une nouvelle formule n'est satisfaite dans aucun des modèles de la base, il faut modifier ces modèles. La distance entre les modèles est évaluée par le *nombre* d'atomes dont la valeur de vérité est modifiée ; le résultat de la révision est alors l'ensemble des formules dont les modèles diffèrent le moins des modèles initiaux. On recherche donc les modèles dans lesquels la formule à ajouter est vérifiée et qui diffèrent minimalement des modèles de la base ; pour cela, on commence par examiner les modèles qui diffèrent d'un atome, puis de deux, etc. Plus formellement, soit ψ un ensemble de formules et μ une formule inconsistante avec ψ, c'est-à-dire qui n'est satisfaite dans aucun modèle de ψ ; *mod*(ψ) est l'ensemble des modèles de ψ. Soit g(i, *mod*(ψ)) l'ensemble des modèles qui diffèrent sur la valeur de vérité d'au plus i atomes. Soit G(i,ψ) l'ensemble de formules tel que : *mod*(G(i,ψ))=g(i, *mod*(ψ)). G(i,ψ) est l'ensemble de formules dont les modèles sont exactement ceux de ψ, modifiés pour au plus i atomes. L'opérateur de révision est alors défini par :

$\psi\circ\mu$ = G(k,ψ)$\cup\{\mu\}$ avec k le plus petit i tel que G(i,ψ)$\cup\{\mu\}$ est consistant.

Cet opérateur satisfait tous les postulats (G1-8) et a la particularité de ne pas utiliser d'opérateur de contraction. Mais il est nécessaire de vérifier la consistance à chaque incrément de *i*, problème NP-complet. Contrairement à [Winslett88], c'est le cardinal des ensembles d'atomes modifiés et non l'inclusion ensembliste qui est pris en compte pour déterminer la minimalité. La différence apparaît dans l'exemple suivant simplifié à l'extrême : soit M un modèle de la base initiale et soient trois modèles M1, M2 et M3 dans lesquels la formule à ajouter est valide. M1 diffère de M par la valeur de vérité d'un atome ; M2 et M3 en diffèrent par la valeur de deux atomes. De plus, l'un des deux atomes modifiés dans M2 est celui modifié dans M1. Si la minimalité est définie par l'inclusion ensembliste, M1 et M3 sont des modèles de la base révisée. Mais si elle est définie grâce aux cardinaux, seul M1 est modèle de la base révisée.

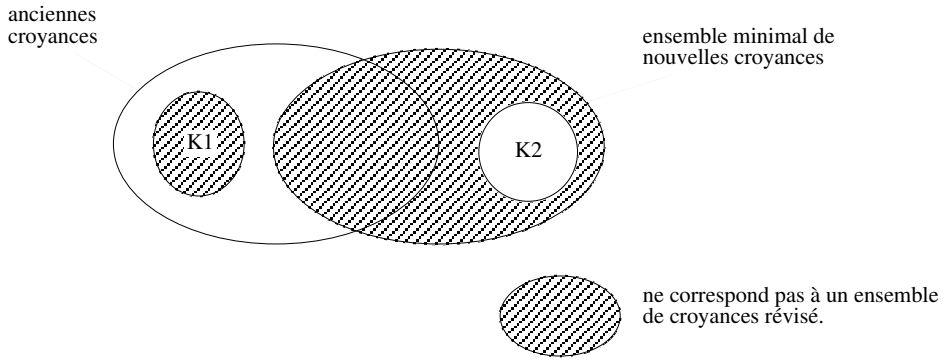


Figure 8 : Les ensembles de croyances révisés minimaux sont tels que pour tout K1 et K2, l'ensemble grisé n'est pas un ensemble de croyances révisé (il ne permet pas de déduire les connaissances indubitables et le dernier ajout).

[Satoh88] s'intéresse à la révision en logique du premier ordre et distingue la connaissance – tout ce qui est connu de manière certaine – des croyances – ce qui est déduit des connaissances et d'autres croyances –, afin qu'une partie de la base soit considérée comme indubitable. Il définit un critère de minimalité qui minimise la différence entre les ensembles de croyances avant et après la révision ; il n'existe pas de base qui conserve plus d'anciennes croyances et en ajoute moins (Figure 8).

[Borgida85] s'occupe des bases de données qui n'évoluent pas dans le cadre rigide habituel ; ainsi, il peut y avoir violation de contraintes d'intégrité ou définition incorrecte d'attribut. Pour définir les révisions minimales, il se base sur les interprétations des atomes comme Dalal, mais il considère l'inclusion ensembliste et non le cardinal des ensembles révisés.

[Katsuno&91a] donne une caractérisation des modifications minimales en se basant sur les modèles. L'idée est de trouver les modèles de la base révisée minimale ($mod(K \circ x)$), c'est-à-dire celle qui satisfait les postulats (K1-6), à l'aide d'un ordre sur les modèles de la

proposition à ajouter : seuls les modèles de la proposition les plus proches des modèles de la base seront modèles de la base révisée. Pour cela, il faut associer à chaque base un ordre sur les interprétations du langage. La fonction qui à chaque base K, représentée par un ensemble de propositions, associe un pré-ordre total $\leq_K$ entre les interprétations du langage, est 'fidèle' (*faithful*) si elle satisfait les contraintes :

- si $I, I' \in \text{mod}(K)$ alors $\neg(I <_K I')$[7]
- si $I \in \text{mod}(K)$ et si $I' \notin \text{mod}(K)$ alors $I <_K I'$
- si $K \equiv K'$ alors $\leq_K = \leq_{K'}$

Si $\leq_K$ représente alors une distance entre une interprétation et les modèles de la base K, $I \leq_K I'$ signifie que I' est plus proche des modèles de K que I, selon l'ordre donné. Le théorème fondamental est alors :

Un opérateur de révision ° satisfait les postulats (K1-6) si il existe une telle fonction 'fidèle' qui satisfait : $\text{mod}(K \circ x) = \min_{\leq_K} \text{mod}(x)$

Ce théorème permet d'étudier en particulier les opérateurs de révision de Dalal, Satoh et Borgida. L'opérateur de Dalal satisfait les postulats (K1-6) car il est possible de définir une fonction fidèle, qui satisfasse le théorème, en basant l'ordre sur le nombre d'atomes dont la valeur de vérité est modifiée entre deux interprétations. Par contre, il n'est pas possible de trouver une telle fonction pour les opérateurs définis par Satoh et Borgida, opérateurs qui ne satisfont pas le postulat (K6).

d) Approche mixte

Ne pas tenir compte de la syntaxe d'une base n'est pas toujours satisfaisant. C'est pourquoi des approches mixtes ont été développées [Papini&94, Papini&95] : d'une part la révision est définie en utilisant les modèles de la base et d'autre part la forme syntaxique de la base permet d'exprimer la préférence entre les modèles. Ils introduisent deux nouveaux opérateurs unaires en logique propositionnelle, appelés bémol et dièse ; le nombre de bémols et de dièses appliqués à une formule indiquent son degré de validité : moins elle a de bémols et plus elle a de dièses, plus elle est valide. Lors de révision itérée de la base, un bémol (resp. un dièse) est ajouté à la base si la dernière formule est jugée par le concepteur plus valide (resp. moins valide) que le contenu précédent de la base. La déduction se définit alors en tenant compte de ces opérateurs, afin de ne déduire que les formules non contradictoires avec les informations les plus sûres.

---

[7] ce qui peut aussi s'exprimer par : $I \leq_K I'$

e) Système de modification de croyances

Afin de représenter explicitement la connaissance qui permet de déterminer comment les croyances sont modifiées quand de nouvelles connaissances sont introduites, [Friedman&95] définit un système de modification de croyances. Ce système a trois composants : l'ensemble des états épistémiques possibles, une fonction qui associe à chaque état épistémique l'ensemble des croyances et une fonction de transition qui détermine comment les états épistémiques sont modifiés suite à l'apprentissage d'une nouvelle connaissance. Un état épistémique correspond à un ensemble ordonné de mondes possibles. La fonction de transition est telle que les nouveaux états épistémiques, suite à l'ajout d'une nouvelle connaissance, sont les mondes qui satisfont la nouvelle connaissance et qui sont le plus proche possible des mondes associés aux états épistémiques initiaux. Friedman et Halpern définissent ainsi une caractérisation logique des opérations de modification de croyances, la révision et la mise à jour en étant deux cas particuliers. Cette caractérisation permet notamment de définir facilement la révision itérée.

3. Conclusion

Il existe donc un ensemble de stratégies permettant de réviser des bases exprimées dans un formalisme logique, la révision satisfaisant des principes tels que : la base révisée doit être consistante, la dernière connaissance est ajoutée dans la base, le maximum de connaissances est conservé. Il existe deux types principaux d'approches : syntaxique (Gärdenfors, Nebel) et sémantique en utilisant les modèles de la base (Dalal, Borgida, Satoh, Friedman).

La plupart des approches proposent un cadre formel sans se soucier de la réalisation pratique. Elles souffrent d'un problème de complexité qui ne permet pas d'envisager une implémentation. [Eiter&92] a étudié la complexité des opérateurs proposés par Dalal, Borgida, Winslett et Satoh. Le premier est $P^{NP(O(\log n))}$ et les trois derniers sont $\prod_p^2$-complet. De même, la révision avec priorité définie par [Nebel92] est $\prod_p^2$-complet, même si le fait de raisonner sur des bases finies permet de concevoir un algorithme. Ces opérateurs de révision ne semblent pas pouvoir être implémentés de manière efficace.

Des compléments sur la théorie de la complexité se trouvent dans [Johnson90].

G. Quelle révision dans les représentations par objets ?

Tout le début de ce chapitre a permis de se rendre compte des possibilités de maintenir la consistance d'une base, que celle-ci soit exprimée en logique propositionnelle, en logique du

premier ordre, fermée par déduction ou non, ou soit une base de données. Il est possible de raisonner avec une base inconsistante ou de proposer des modifications de la base afin qu'elle reste consistante. L'idée est toujours d'utiliser des sous-ensembles de la base qui soient maximaux selon l'inclusion ensembliste, cette notion de maximum utilisant éventuellement un ordre introduit entre les connaissances de la base.

Étudions à présent les besoins dans les représentations de connaissance par objets.

1. Quels sont les besoins ?

a) Besoin d'une aide pour préserver la consistance d'une base

Les bases de connaissance exprimées dans un formalisme par objets, comme celui décrit à la section I.C, font face aux mêmes inconsistances potentielles que toute autre base. Les concepteurs ne sont pas en mesure de garantir que la dernière connaissance entrée dans la base est consistante avec le reste de la base. Cependant, la base doit permettre de représenter leur vision d'un certain domaine qui, lui, est consistant (tout simplement car il existe). Il s'agit donc de garantir que le contenu de la base est consistant afin de pouvoir l'utiliser en toute confiance, en particulier obtenir des inférences (inférences de valeurs, classification, etc.) qui soient valides dans le domaine. Cette validité dépend également de la modélisation qui a été faite du domaine, mais la consistance est une condition nécessaire.

Si une nouvelle connaissance est consistante avec la base de connaissance, elle est simplement ajoutée à la base. Rappelons en effet que l'hypothèse d'un domaine immuable est posée, et donc que la problématique diffère de celle de la mise à jour qui, dans ce cas, conduit à plus modifier la base que par le simple ajout (voir paragraphe E.5.c).

Si l'ajout de la nouvelle connaissance conduit à une base inconsistante, la base doit être réduite afin d'être consistante avec la dernière connaissance : il s'agit de la *réviser*. Nous adoptons à partir de maintenant une définition de la révision plus restreinte que précédemment : on ne parlera de révision que lors de l'ajout d'une connaissance *inconsistante avec la base.*

Pour un concepteur de base, il est relativement difficile d'appréhender l'ensemble du contenu de la base et donc d'identifier les modifications de la base qui permettent la révision suite à un ajout inconsistant. Il faut donc automatiser ce processus en proposant au concepteur les différentes possibilités (Figure 9). Seules les modifications permettant d'obtenir des bases révisées de manière minimale, c'est-à-dire contenant le maximum de connaissances de la base initiale, sont à prendre en compte. Cette notion de minimalité sera détaillée par la suite. D'autre part, les modifications proposées peuvent porter sur le contenu de la base (comme en révision dans les systèmes logiques), mais également sur l'ajout lui-même. Nous verrons que ces

propositions spécifiques aux langages de représentation par objets sont importantes et relativement intuitives.

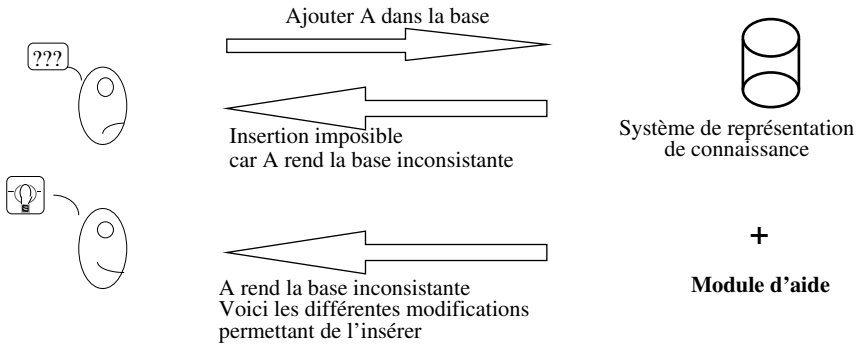


Figure 9 : Apport d'un module d'aide pour un concepteur d'une base de connaissance : le mécanisme de révision va être interactif ; les différentes révisions possibles de la base seront proposées au concepteur.

Afin de résoudre les inconsistances soulevées par la dernière assertion, il est donc proposé au concepteur de (Figure 10) :

réviser la base : le concepteur veut que son assertion appartienne à la base et il s'agit donc de réviser la base de manière minimale. Nous allons décrire un mécanisme de révision adaptée à une représentation par objets, ainsi que l'algorithme associé.

modifier son assertion : le concepteur souhaite modifier son assertion après s'être rendu compte qu'elle est inconsistante. Il peut soit la modifier afin qu'elle soit consistante avec la base, soit la modifier de façon moindre et réviser ensuite la base avec cette nouvelle assertion. Ceci sera étudié à la section IV.D ;

rejeter son assertion : le concepteur se rend compte que son assertion n'est pas acceptable et ne tient alors pas à la conserver dans la base ; il privilégie le contenu de la base à sa dernière assertion.

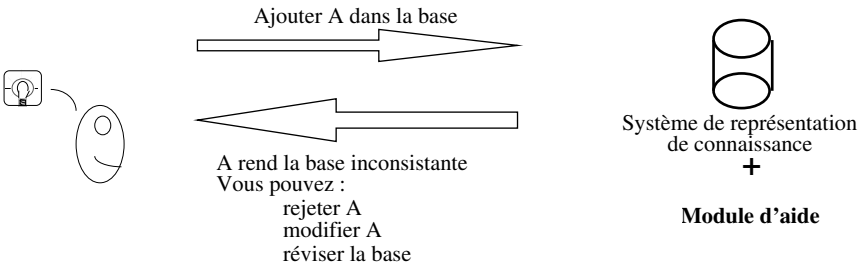


Figure 10 : Suite à l'ajout d'une connaissance qui rend la base inconsistante, il est possible de rejeter l'assertion, de la modifier ou de réviser la base.

b) Besoin d'une base consensuelle

L'application de ce travail se situe dans une optique plus large que la construction d'une base de connaissance par un concepteur isolé. [Euzenat95] étudie la construction coopérative de bases de connaissance consensuelles grâce à une organisation hiérarchique de bases :

chaque concepteur développe sa propre base et participe à la construction d'une base consensuelle avec d'autres concepteurs en soumettant les connaissances de sa base qu'il juge pertinentes pour le groupe ; plusieurs groupes de concepteurs peuvent exister, chacun créant sa base consensuelle ; ces bases peuvent ensuite être rassemblées dans une base consensuelle à un niveau supérieur, etc. (Figure 11).

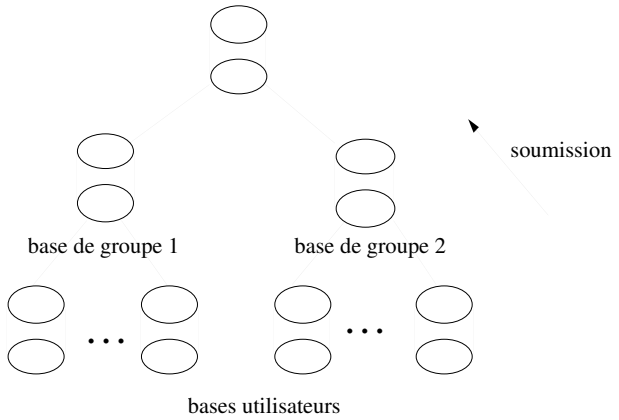


Figure 11 : Organisation hiérarchique de bases. Les bases de groupe regroupent la connaissance consensuelle de toutes leurs sous-bases.

L'objectif est de favoriser le travail coopératif afin d'aboutir à une base consensuelle tout en permettant à chaque concepteur de préserver ses hypothèses. Dans ce processus de construction, des inconsistances peuvent évidemment apparaître lors de la soumission de connaissances si des concepteurs ne sont pas d'accord sur un point donné, d'où l'utilité d'un mécanisme de révision de base suite à l'ajout d'une connaissance inconsistante.

Ce besoin de construire des bases de connaissances consensuelles est visible dans deux domaines d'application de TROPES. D'une part, en biologie moléculaire, il est intéressant de pouvoir regrouper toutes les connaissances disponibles sur un gène particulier par exemple. D'autre part, pour la mémoire d'entreprise, une base de connaissance permet la mise en commun d'un ensemble de connaissances sur l'organisation de l'entreprise, ses produits, ses méthodes de conception, etc. La construction étant le fruit du travail de plusieurs concepteurs, il est vital qu'il existe un processus permettant la mise en commun et la vérification de la cohérence de l'ensemble des connaissances.

2. Une révision spécifique aux représentations de connaissance par objets

La révision logique ne se révèle pas satisfaisante pour réviser les bases de connaissance par objets (paragraphe a), ce qui requiert de tirer parti des particularités des représentations de

connaissance par objets (paragraphe b). De plus, il est nécessaire de définir un langage formel de représentation par objets (paragraphe c).

a) Utilisation de la révision logique ?

Les postulats posés pour la révision en logique correspondent aux attentes pour la révision dans les représentations de connaissance par objets (RCO). En effet, comme cela a été précisé partiellement dans l'analyse des besoins (paragraphe 1.a), la révision dans les RCO doit être telle que :

- toute base révisée est consistante,
- la dernière connaissance appartient aux bases révisées,
- le maximum de connaissances est conservé,
- aucune connaissance superflue n'est ajoutée,
- toutes les bases révisées possibles sont traités équitablement,
- l'algorithme est efficace.

Ces contraintes se retrouvent dans les principes donnés pour la révision (paragraphe F.1.b). Les travaux de la révision en logique sont immédiatement applicables aux objets en traduisant les représentations par objets dans un formalisme logique car il est possible de découper une base de connaissance en un ensemble de formules logiques [Crampé&96]. Cependant, les différentes stratégies de révision en logique souffrent de problème de complexité (notamment pour la vérification de la consistance) et il n'est donc pas possible de définir des algorithmes permettant d'implémenter efficacement la révision. Or, nous souhaitons obtenir un mécanisme de révision qui soit utilisable concrètement dans un système de représentation de connaissance par objets, tout en satisfaisant les principes énoncés plus tôt et notamment ceux donnés par [Alchourròn&85].

De plus, le résultat de la révision logique est *une* base, ce qui nécessite un choix entre les différentes bases révisées possibles en faisant intervenir des préférences entre les connaissances ou entre les modèles. Or, nous n'envisageons pas de demander aux concepteurs de bases de connaissance de fournir de telles informations, car cela alourdirait considérablement la construction des bases. De plus, les concepteurs devraient avoir une vue globale de la base, ce qui est contraire à notre hypothèse. Cependant, parmi l'ensemble des parties de la base qui sont consistantes avec la dernière connaissance qu'un concepteur souhaite ajouter, celles qui font des suppressions superflues de connaissances ne sont pas à garder.

Ces deux remarques impliquent la nécessité de définir une révision spécifique aux RCO.

b) Spécificité de la révision souhaitée

Définir une révision dans les RCO nécessite de résoudre trois problèmes :

- obtenir une complexité moindre que la révision logique,
- adapter la révision aux RCO pour que l'inconsistance et la minimalité notamment s'interprètent de manière intuitive,
- gérer le fait que la minimalité ne peut conduire à une seule base révisée.

L'idée est d'utiliser les multiples entités (classes, objets, attributs) des modèles à objets et des règles d'inférence adaptées pour guider la recherche des inconsistances et des bases révisées. L'inconsistance est alors caractérisée de manière spécifique aux représentations par objets, caractérisation qui permet de la détecter et de la localiser facilement lors d'un ajout. En outre, la notion d'incohérence est introduite pour refléter un problème spécifique aux bases de connaissance à objets, à savoir l'existence de classes ne pouvant contenir d'objet ; ce genre de problème est à régler également lors de la révision.

De plus, la notion de minimalité, qui est définie selon le principe de préservation du maximum de connaissance, peut aussi s'interpréter intuitivement en utilisant l'ordre de la relation de spécialisation. Ainsi, si la révision d'une base nécessite de changer un objet de classe, il sera déplacé vers la classe la plus proche possible de sa classe d'appartenance initiale. Puisque les connaissances ne sont pas ordonnées, cette notion de minimalité conduit cependant à plusieurs bases révisées. Le choix parmi elles ne peut être guidé que par la détermination des suppressions des connaissances jugées les moins importantes par le concepteur. C'est pourquoi toutes ces bases révisées minimales lui sont proposées afin qu'il choisisse. Comme ce nombre peut être important, l'algorithme sera défini de telle sorte que la révision se fasse pas à pas, ciblant ainsi à chaque étape les connaissances privilégiées par rapport à d'autres.

Grâce à l'utilisation de la structure des bases de connaissance et à l'interactivité, le coût pour obtenir la révision d'une base est polynomial.

c) Besoin d'une représentation formelle

Afin de décrire le mécanisme de révision dans une représentation par objets, il est nécessaire de doter le système de RCO d'une formalisation. En effet, la syntaxe du langage doit être définie afin qu'une base puisse être représentée. De plus, afin de donner à la base toute sa dimension de "modèle d'un domaine", il est nécessaire de définir la sémantique du langage de représentation, afin de préciser le lien qui existe entre les formules contenues dans la base et les entités du domaine. Les mécanismes de déduction doivent ensuite être définis, afin de pouvoir obtenir les conséquences de la base sans utiliser les modèles. Enfin, nous avons vu qu'il n'est pas possible d'utiliser de manière pratique la clôture déductive des bases,

et il est donc nécessaire que l'opérateur de révision soit défini grâce à une représentation *finie* des bases.

Notre objectif est donc de proposer un mécanisme de révision pour les bases de connaissance par objets, en se basant sur leurs particularités (instanciation, attributs, relation de spécialisation entre classes), ce qui nécessite en premier lieu la définition d'un langage d'objets.

Dans la suite, nous allons tout d'abord décrire le langage d'objets en commençant par un langage d'objets relativement restreint, complexifié dans un second temps (chapitre II). Nous définissons alors la révision et la notion de minimalité pour les bases de ce langage (chapitre III). Nous présentons enfin un algorithme permettant de donner les bases révisées suite à un ajout inconsistant (chapitre IV).

Chapitre II. Formalisation d'un langage d'objets

Nous avons vu qu'il est capital de définir formellement le langage d'objets dans lequel sont exprimées les bases qui seront révisées. Cette définition est nécessaire car il n'existe pas – à notre connaissance – de logique d'objets qui ait une sémantique conforme à celle qui est souhaitée.

En effet, la formalisation en théorie des types [Abadi&96] ne développe pas une sémantique dénotationnelle simple qui interprète les classes comme des ensembles d'individus d'un domaine. Dans la F-logic définie par [Kifer&95], l'interprétation se fait dans un domaine supposé déjà structuré : les classes sont interprétées par des classes du domaine et la spécialisation est interprétée comme une relation existante. Or, nous souhaitons interpréter une base dans un domaine qui se compose uniquement d'un ensemble d'entités décrites par leurs caractéristiques ; les classes sont alors interprétées par des ensembles d'entités et la spécialisation par l'inclusion ensembliste. Enfin, la formalisation des objets sous forme de systèmes classificatoires [Ducournau96b] aurait pu être utilisée, mais elle n'avait pas atteint son niveau de développement actuel au moment du début de ce travail ; en particulier, il n'existait pas de liens complets avec la théorie des modèles, ni de système déductif.

De plus, la définition d'un nouveau langage permet de se restreindre au premier abord à un langage simple, qui est complexifié par la suite. Présenter le langage simple a pour seul objectif de rendre plus compréhensible la description du langage plus complexe.

La première partie décrit ce langage simple et la seconde se focalise sur les compléments qui doivent être apportés afin de décrire une base de connaissance acceptant les liens entre les objets. Une troisième partie indique les différences entre le langage décrit et le langage terminologique et TROPES.

Cette première partie ainsi que le descriptif de la révision adaptée à ce langage ont fait l'objet de [Crampé&96b] et d'un rapport de recherche [Crampé&96c]. Les preuves des propositions de la première partie peuvent y être trouvées ; elles sont plus spécifiques que celles données en annexe B qui correspondent au langage complété présenté en seconde partie.

A. Un langage d'objets simple

Cette partie définit un langage restreint de représentation de connaissance par objets. En particulier, les liens entre instances ne sont pas autorisés, ou en d'autres termes les domaines des attributs sont restreints aux types de base.

Pour fixer le formalisme sur lequel la révision s'applique, le langage d'expression des objets est défini (paragraphe 1). Il est doté d'une sémantique correspondant à ce qui est attendu des représentations par objets considérées ; les notions de modèle et consistance sont définies (paragraphe 2). Un ensemble de règles d'inférence est proposé pour ce langage. Il permet d'assurer la complétude avec la sémantique ; de plus, il existe alors des procédures de déduction polynomiales (paragraphe 3). Une forme normalisée des bases de connaissance, nommée forme première et utilisée pour la révision, est ensuite présentée : c'est une description de la base par un ensemble fini minimal de connaissances (paragraphe 4).

Un index des notations utilisées et un sommaire des définitions se trouvent en annexe A.

1. Syntaxe

La syntaxe est découpée en deux parties. Une première partie correspond à la grammaire donnant l'expression des entités de la base et à l'énoncé de contraintes méta-syntaxiques. La seconde permet de considérer des types de données qui peuvent être ajoutés au système : entiers, booléens, ainsi que tout autre type défini par un type abstrait.

a) Grammaire du langage

Le langage permet de décrire des classes en relation de spécialisation, des instances de ces classes et des attributs qui sont typés dans les classes par un domaine et qui sont valués dans les instances par une valeur. La notion de consistance étudiée au paragraphe 2.b permettra de définir les bases correctes[8].

La Figure 12 est la représentation graphique d'une base, contenant cinq classes (c_0, c_1, c_2, c_3, c_4), trois instances i, i' et i'' rattachées respectivement à c_1, c_3 et c_4 et un seul attribut a[9].

[8] La définition détaillée des bases problématiques est faite en section III.A dans le chapitre concernant la révision.

[9] Les exemples donnés utiliseront volontairement des noms de classes et d'objets détachés de toute réalité, afin de pouvoir les exploiter indépendamment de tout contexte et d'en donner des interprétations différentes.

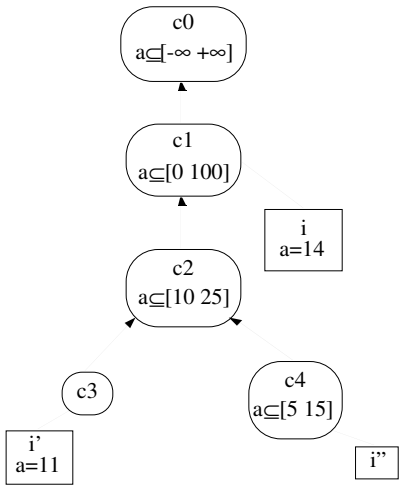


Figure 12 : exemple de base de connaissance représentée graphiquement.

Une base de connaissance est définie comme étant un ensemble d'assertions, une assertion exprimant :

- la relation de spécialisation, notée ≤, liant une classe à sa super-classe ($c_1 \leq c_0$) ;
- le rattachement, noté ∈, d'une instance à une classe ($i \in c_2$) ;
- la restriction d'un attribut d'une classe par un certain domaine ($c_1.a \subseteq [0\ 100]$) ;
- ou la valeur d'un attribut d'une instance ($i.a=14$).

Ceci est repris ci-dessous par la définition de la grammaire de la base.

DÉFINITION 1 : GRAMMAIRE

<base_de_connaissance> ::= {<assertion>*}
<assertion> ::= <relation_spécialisation> | <rattachement_instance> | <restriction_attribut_classe> | <valeur_attribut_instance>
<relation_spécialisation> ::= <nom_classe> ≤ <nom_classe>
<rattachement_instance> ::= <nom_instance> ∈ <nom_classe>
<restriction_attribut_classe> ::= <nom_classe>.<nom_attribut> ⊆ <domaine>
<valeur_attribut_instance> ::= <nom_instance>.<nom_attribut> = <valeur>
<nom_classe> | <nom_instance> | <nom_attribut> ::= <identificateur>

Les <identificateur> sont des chaînes alphanumériques quelconques. <domaine> et <valeur> sont définis ci-dessous. De plus, un certain nombre de contraintes méta-syntaxiques doivent être ajoutées :

- Une base de connaissance est un ensemble *fini* d'assertions.
- Il est possible d'introduire plusieurs fois les classes et les instances : la multi-spécialisation, qui indique pour une classe la possibilité d'être sous-classe de plusieurs

classes incomparables, est autorisée ; la multi-instanciation, qui indique la possibilité pour une instance d'être attachée à plusieurs classes incomparables, est également autorisée.

- Le graphe de la relation ≤, appelée spécialisation, entre les <nom_classe> est sans circuit.
- Les assertions de la forme ($c' \leq c$), ($o \in c$) ou ($c.a \subseteq d$) peuvent être ajoutées dans la base même si la classe c n'a pas été introduite par une assertion de la forme ($c \leq c''$).
- Une valeur d'attribut d'instance peut utiliser un attribut qui n'a pas été introduit par une assertion de restriction d'attribut de classe.
- La valeur d'attribut d'une instance ne peut être une autre instance. Ceci est la limitation principale dans cette première partie, qui est levée dans la seconde partie.
- La multi-valuation, c'est-à-dire la faculté d'avoir plusieurs valeurs pour un attribut d'instance est autorisée par la syntaxe.
- Il n'existe aucune contrainte particulière entre le domaine d'un attribut d'une classe et celui de ses super-classes[10]. Ainsi, dans la base de la Figure 12, c_4 est une sous-classe de c_2 ($c_4 \leq c_2$), mais $c_4.a \subseteq [5\ 15]$, $c_2.a \subseteq [10\ 25]$ et [5 15] n'est pas inclus dans [10 25].
- Aucune contrainte n'est posée entre la valeur d'un attribut d'instance et les restrictions de domaine des classes auxquelles elle est attachée[11].

Selon la grammaire donnée, la base de connaissance de la Figure 12 s'exprime syntaxiquement par :

$B=\{c_1 \leq c_0\ ;\ c_2 \leq c_1\ ;\ c_3 \leq c_2\ ;\ c_4 \leq c_2\ ;\ i \in c_1\ ;\ i' \in c_3\ ;\ i'' \in c_4\ ;\ c_0.a \subseteq [-\infty\ +\infty]\ ;\ c_1.a \subseteq [0\ 100]\ ;\ c_2.a \subseteq [10\ 25]\ ;\ c_4.a \subseteq [5\ 15]\ ;\ i.a=14\ ;\ i'.a=11\}$.

b) Notations

Dans la suite, c, c', c_1,… dénoteront des classes, i, i', o, o' des instances, a, a' des attributs, v, v' des valeurs et d, d' des domaines. Les assertions sont de l'une des quatre formes $c \leq c'$, $o \in c$, $c.a \subseteq d$ ou $o.a=v$. De plus, l'appartenance d'une assertion, de la forme ($o \in c$) par exemple, à une base B est notée $(o \in c) \in B$.

Pour $c, c' \in C$, $(c \leq^* c') \in B$ dénote :

$(c \leq c') \in B$ ou

il existe $c'' \in C$ telle que $(c \leq^* c'') \in B$ et $(c'' \leq c') \in B$.

---

[10] Cependant, ceci n'est vrai qu'au niveau syntaxique, car l'interprétation de la relation de spécialisation conduira à ce que la restriction du domaine d'un attribut de classe soit l'intersection des restrictions posées dans toutes les super-classes.

[11] Mais ceci sera précisé lors de la définition de l'interprétation de la base et notamment des modèles de la base (paragraphe 2)

DÉFINITION 2 : ENSEMBLES D'ENTITÉS

Les ensembles C_B, O_B, A_B sont respectivement les ensembles de classes, instances et attributs introduits dans la base B par une assertion les faisant intervenir.

Les ensembles V_B et T_B sont respectivement les ensembles des valeurs et des domaines des types introduits dans la base B.

Ainsi, l'ensemble C_B contient toutes les classes c telle qu'elles apparaissent dans une assertion de la base de la forme $c \leq c'$, $c' \leq c$, $c.a \subseteq d$ ou $o \in c$.

c) Opérations sur les types et les domaines

Un type définit un ensemble de valeurs et des opérations sur celles-ci. Suivant les caractéristiques du type (ordonné, fini, etc.), les domaines destinés à représenter des sous-ensembles de valeurs du type ont une représentation différente (intervalles, vecteurs de bits, etc.) et les opérations applicables aux valeurs (égalité, comparaison, etc.) sont différentes. Pour chaque type τ, on suppose qu'il existe un langage de valeurs V^τ et un langage de domaine T^τ disjoints des ensembles C_B, O_B et A_B. L'union des V^τ (resp. des T^τ) forment V_B (resp. T_B).

Les opérations suivantes sur les domaines et les valeurs sont utilisées [Capponi95] ; elles sont indicées par le type τ des domaines ou des objets concernés. Soient d, d' des domaines et v, v' des valeurs du type τ.

$d \wedge_\tau d'$: intersection des domaines de valeurs d et d' ;
$v :_\tau d$ ($\neg v :_\tau d$) : la valeur v appartient (resp. n'appartient pas) au domaine d ;
$d =_\tau d'$ ($d \neq_\tau d'$) : le domaine d est égal au (resp. différent du) domaine d' ;
$d \leq :_\tau d'$ ($d <:_\tau d'$) : le domaine d est inclus (resp. strictement) dans le domaine d' ;
$v \leq_\tau v'$: la valeur v est inférieure à la valeur v', τ étant ordonné ;
$v =_\tau v'$ ($v \neq_\tau v'$) : la valeur v est égale à (resp. différente de) la valeur v'.

$\perp_\tau$ est le symbole représentant le domaine vide du type τ.

Le symbole τ sera omis là où il ne sera pas nécessaire.

Dans la suite, on ne considère qu'un seul type, celui des entiers : les valeurs sont restreintes aux entiers relatifs et les domaines aux unions finies d'intervalles d'entiers relatifs.

<valeur> : := <entier>
<domaine> : := <liste_intervalles> | ⊥
<liste_intervalles> : := <intervalle> ∪ <liste_intervalles> | <intervalle>
<intervalle> : := [<borne⁻> <borne⁺>]
<borne⁻> : := <entier> | -∞
<borne⁺> : := <entier> | +∞

$\perp$ représente le domaine vide. V_B est restreint à l'ensemble des entiers et T_B aux unions finies d'intervalles d'entiers.

2. Sémantique

La syntaxe du langage permet de définir les assertions qui sont acceptées ; afin de pouvoir donner un sens à ces assertions, il faut les relier au domaine que la base doit modéliser. La sémantique est définie classiquement par une fonction d'interprétation des instances vers un domaine en tenant compte des types utilisés dans la syntaxe. Les notions de modèle, conséquence et consistance sont définies ci-dessous.

a) Interprétation

L'interprétation se définit pour les entités de la base, mais aussi pour les valeurs et domaines. L'interprétation de type doit être telle que les opérateurs associés au type aient leur sens naturel : un domaine est plus petit qu'un autre si et seulement si son interprétation est contenu dans l'interprétation du second ; une valeur appartient à un domaine si et seulement si l'interprétation de la valeur appartient à l'interprétation du domaine, etc.

DÉFINITION 3 : INTERPRÉTATION DE TYPE

Soit un type τ, une interprétation des expressions, c'est-à-dire des domaines et valeurs du type τ est une fonction $I_\tau : V_B \to \tau$ et de $T_B \to 2^\tau$ satisfaisant les condition suivantes :

$$I_\tau(\perp_\tau) = \emptyset \qquad I_\tau(d \wedge_\tau d') = I_\tau(d) \cap I_\tau(d')$$

$$d <:_\tau d' \Leftrightarrow I_\tau(d) \subset I_\tau(d') \qquad d \leq:_\tau d' \Leftrightarrow I_\tau(d) \subseteq I_\tau(d')$$

$$d =_\tau d' \Leftrightarrow I_\tau(d) = I_\tau(d') \qquad v \leq_\tau v' \Leftrightarrow I_\tau(v) \leq I_\tau(v')$$

$$v :_\tau d \Leftrightarrow I_\tau(v) \in I_\tau(d) \qquad \neg v :_\tau d \Leftrightarrow I_\tau(v) \notin I_\tau(d)$$

$$v \neq v' \Leftrightarrow I_\tau(v) \neq I_\tau(v') \qquad v = v' \Leftrightarrow I_\tau(v) = I_\tau(v').$$

Vu la restriction des types aux entiers, un domaine est interprété par un intervalle d'entiers et une valeur par un entier ; la fonction d'interprétation de type sera notée I. L'interprétation des domaines peut alors être précisée :

$$I([v\ v'] \cup d) = I([v\ v']) \cup I(d)$$

$$I([v\ v']) = \{v'' \in \tau, \text{ tel que } I(v) \leq_\tau v'' \leq_\tau I(v')\} \text{ (avec } \forall v \in \tau,\ I(-\infty) <_\tau v <_\tau I(+\infty))$$

La fonction d'interprétation de type sera considérée ici comme fixée : il n'en existe qu'une, donnée par l'implémentation, qui correspond aux intervalles sur les entiers relatifs. Il n'en sera donc plus question.

La fonction d'interprétation va permettre de définir la relation entre la base et un domaine particulier. Les domaines sont considérés comme des ensembles d'entités décrites par des couples (caractéristique, valeur). Ainsi, le domaine des individus est constitué d'un ensemble de personnes décrites par leur numéro de sécurité sociale, leur profession, leur adresse, etc. Aucune hypothèse n'est faite sur l'existence de classes dans le domaine représenté qui ne possède donc aucune structure.

Au contraire, dans la F-logic [Kifer&95], l'interprétation se définit sur un domaine contenant classes et objets, les objets appartenant aux classes, celles-ci étant elles-mêmes reliées entre elles par relation de spécialisation. Ainsi, dans le domaine des individus, il existe déjà des classes étudiants ou troisième âge, et on connaît pour un individu particulier les classes auxquelles il est attaché. Ceci limite donc l'utilisation d'une telle logique à des domaines qui sont structurés, alors que l'un des intérêts de la construction d'une base de connaissance est justement d'aider à la structuration des connaissances, par la détermination de classes, des liens de spécialisation entre classes et du rattachement des objets aux classes.

Dans notre cadre, l'interprétation d'une base est donc définie dans un domaine qui est un ensemble d'éléments, chacun étant décrit par un ensemble de couples (caractéristique, valeur).

DÉFINITION 4 : INTERPRÉTATION DES ENTITÉS D'UNE BASE

Soit une base de connaissance B et un domaine D quelconque, disjoint de τ. Une interprétation est un couple $<D,I_B>$, I_B étant appelée fonction d'interprétation :

$I_B : O_B \rightarrow D$ injective sur O_B

$C_B \rightarrow 2^D$

$A_B \rightarrow (D \rightarrow \tau)$ $I_B(a)$ est une fonction totale

$V_B \rightarrow \tau$

$T_B \rightarrow 2^\tau$

telle que I_B restreinte à V_B et T_B est confondue avec I.

La fonction d'interprétation d'une base associe donc à une instance un élément du domaine. L'injectivité de la fonction sur O_B signifie que l'hypothèse de nom unique pour les éléments du domaine est satisfaite : chaque élément du domaine n'est l'interprétation que d'au plus une instance de la base, ou encore deux instances différentes de la base représentent toujours deux entités différentes du domaine. Cette fonction est totale car toutes les instances de la base sont interprétées par une entité du domaine, mais elle n'est pas bijective car le domaine peut contenir des éléments qui ne sont pas représentés dans la base.

La fonction d'interprétation associe à une classe un ensemble d'éléments du domaine et à un attribut, une fonction du domaine vers le type. Pour un attribut donné, la fonction $I_B(a)$ associe

à un élément du domaine une valeur du type ; alors que la multi-valuation est autorisée par la syntaxe du langage, elle est interdite par la sémantique des attributs qui sont des fonctions à valeur dans le type. Cette fonction $I_B(a)$ associe à l'interprétation d'une classe un sous-ensemble du type ou, autrement dit, elle associe à un ensemble d'éléments du domaine un ensemble de valeurs du type.

EXEMPLE

La base prise comme exemple peut être interprétée dans le domaine de toutes les villes du monde (Figure 13).

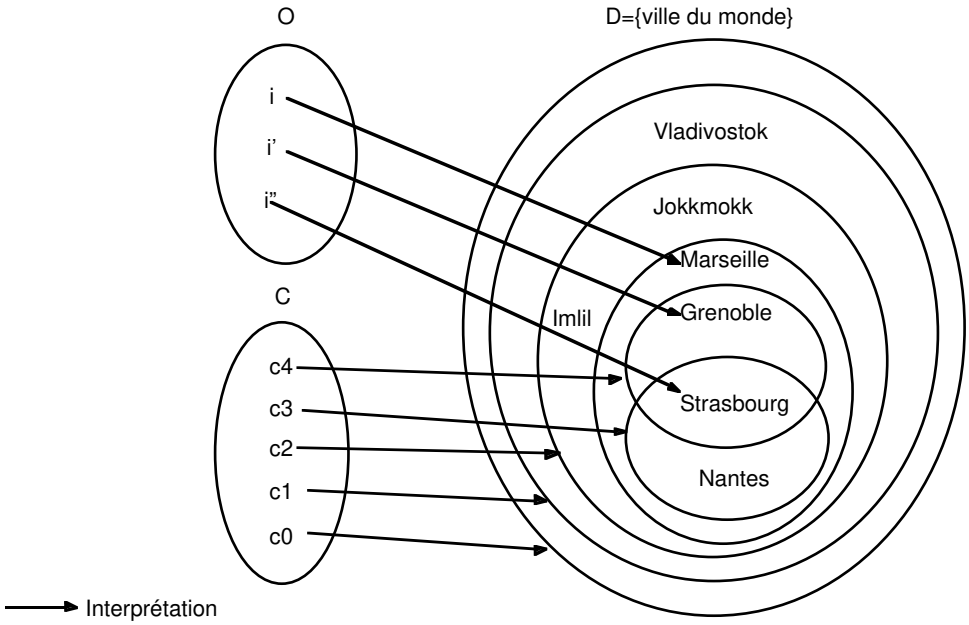


Figure 13 : Les instances et classes de la base de connaissance de la Figure 12 sont interprétées dans le domaine de l'ensemble des villes du monde. Chaque instance est interprétée par une ville, chaque classe comme un ensemble de villes.

L'attribut *a* est interprété par "la température moyenne annuelle de la ville", qui est une fonction du domaine vers les entiers. Les classes sont interprétées par des ensembles de villes.

D={ville du monde}

$I_B : O_B \rightarrow D$

$o \rightarrow$ une ville

$I_B : C_B \rightarrow 2^D$

$c \rightarrow$ {ville}

$I_B : A_B \rightarrow (D \rightarrow \tau)$

$a \rightarrow$({ville du monde}$\rightarrow$entiers)

ville$\rightarrow$température moyenne annuelle

Il est évident que pour une base donnée, s'il existe une interprétation, il en existe un nombre infini, vu qu'il existe un nombre infini de domaines isomorphes au domaine de l'interprétation trouvée. Dans l'exemple traité, le domaine d'interprétation peut être un ensemble de personnes et l'attribut *a* l'âge de ces personnes, un ensemble d'animaux et le poids de leur nourriture journalière, un ensemble de doctorants et leur nombre d'heures de travail journalier, etc.

NOTATIONS

Par la suite, l'indice *B* est omis lorsqu'il n'y a pas d'ambiguïté.

Pour un attribut *a* et une classe *c*, $I(a|c)$ est l'image de $I(c)$ par $I(a)$ et $I(a|c) \subseteq I(d)$ exprime que $I(a|c)$ est une fonction de $I(c)$ dans $I(d)$.

Dans l'exemple précédent (Figure 13), $I(a|c_1)$ est l'image de $I(c_1)$ ={Imlil, Jokkmokk, Marseille, Nantes, Grenoble, Strasbourg} par la fonction $I(a)$="température annuelle moyenne", soit l'ensemble {20, 0, 14, 11, 11, 10}[12].

b) Modèle d'une base, conséquence et consistance

La fonction d'interprétation indique comment les entités de la base (instances, classes, attributs) sont reliées aux entités du domaine. Mais quel sens donner aux assertions ? Il s'agit tout d'abord de voir quelles sont les assertions qui sont satisfaites par une interprétation donnée, c'est-à-dire celles qui sont vraies dans le domaine de l'interprétation. La satisfaction d'une assertion est définie naturellement en indiquant que :

- une classe est sous-classe d'une autre si toutes ses instances sont également des instances de sa super-classe ;
- un attribut d'une classe est restreint par un domaine si la restriction de l'interprétation de l'attribut à la classe est inclus dans l'interprétation du domaine ;
- une instance appartient à une classe si l'interprétation de l'instance appartient à l'ensemble d'objets qui est interprétation de la classe ;
- un attribut d'une instance possède une valeur donnée si l'image de l'interprétation de l'instance par l'interprétation de l'attribut est l'interprétation de la valeur.

Ces points sont repris formellement dans la définition suivante.

DÉFINITION 5 : SATISFACTION D'UNE ASSERTION

Une interprétation $<D,I>$ qui satisfait une assertion δ est noté $\models_{<D,I>} \delta$ et

[12] Ces informations sont tirées du serveur http://www.WorldClimate.com.

$\models_{<D,I>} c \leq c'$ ssi $I(c) \subseteq I(c')$

$\models_{<D,I>} c.a \subseteq d$ ssi $I(a \mid c) \subseteq I(d)$

$\models_{<D,I>} o \in c$ ssi $I(o) \in I(c)$

$\models_{<D,I>} o.a = v$ ssi $I(a)(I(o)) = I(v)$

La négation de la satisfaction est notée $\not\models$.

EXEMPLE

Dans l'exemple traité dans la Figure 13, l'assertion $(i.a=14)$ implique que $I(a)(I(i))=I(14)$, ce qui signifie que Marseille est une ville dont la température moyenne est de 14°. Réciproquement, vu que Grenoble appartient à l'interprétation de la classe c_4, on a $\models_{<D,I>} i' \in c_4$.

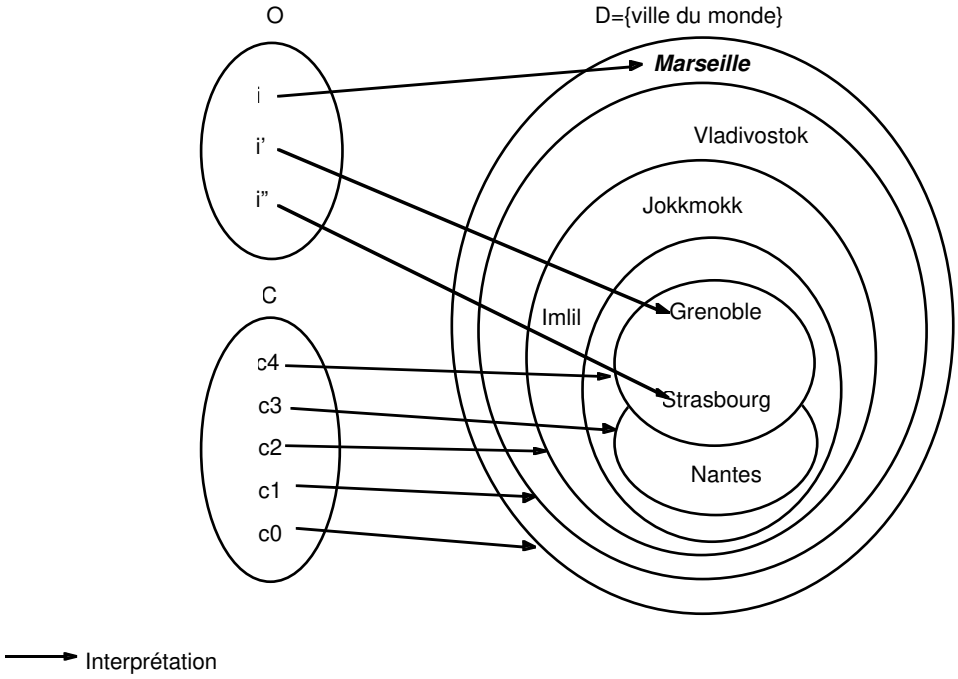


Figure 14 : Exemple d'interprétation de la base qui n'est pas un modèle. En effet, $(i \in c_1) \in B$ mais $\not\models_{<D,I>} i \in c_1$ car $I(i) \notin I(c_1)$, puisque Marseille n'appartient pas à l'interprétation de la classe c_1.

Il est alors possible de définir les interprétations qui sont des modèles de la base, c'est-à-dire qui satisfont toutes les assertions de la base.

DÉFINITION 6 : MODÈLE

$M=<D, I>$ est un modèle d'une base de connaissance B, noté $\models_M B$, ssi M satisfait toutes les assertions de la base B.

L'interprétation donnée en exemple par la Figure 13 est un modèle car toutes les assertions de la base sont satisfaites. Tel n'est pas le cas de l'interprétation donnée par la Figure 14.

La définition des satisfactions d'assertion introduit des contraintes sur la fonction d'interprétation d'un modèle d'une base, qui correspondent à ce qui est attendu d'un modèle à objets :

- L'interprétation d'une classe doit être incluse dans l'interprétation de chacune de ses super-classes et dans l'ensemble d'éléments du domaine qui satisfont les restrictions de ses attributs. Si aucune restriction n'est posée sur les attributs de la classe, son interprétation est simplement incluse dans *D*.
- L'interprétation d'une instance doit appartenir à l'interprétation de chacune des classes à laquelle elle est attachée et à l'ensemble des éléments du domaine qui satisfont les restrictions posées sur la valeur de ses attributs ; ceci revient à dire que l'interprétation d'une instance doit satisfaire les restrictions d'attribut imposées dans ses classes d'appartenance.
- Le co-domaine de la restriction de l'interprétation d'un attribut à une classe doit être inclus dans celui de ce même attribut pour les super-classes et dans l'interprétation des domaines qui restreignent l'attribut pour cette classe particulière. Si aucune restriction n'est posée sur un attribut, son domaine est simplement inclus dans τ.
- L'interprétation d'un attribut d'une instance doit appartenir à l'interprétation de l'attribut de ses classes et aux interprétations des valeurs données pour cet attribut[13].

Les contraintes sur chacun des éléments sont synthétisées dans la proposition suivante :

PROPOSITION 1 :
Si $M=<D,I_B>$ est un modèle de *B*, alors :

$$I_B(c) \subseteq D \cap \bigcap_{(c \leq c') \in B} I_B(c') \cap \bigcap_{(c.a \subseteq d) \in B} \{e \in D; I_B(a)(e) \in I_B(d)\}$$

$$I_B(o) \in \bigcap_{(o \in c) \in B} I_B(c) \cap \bigcap_{(o.a=v) \in B} \{e \in D; I_B(a)(e) = I_B(v)\}$$

Les fonctions dans $I(T_A^B)$ sont covariantes, c'est-à-dire que si $I(c) \subseteq I(c')$ alors $I(a|c) \subseteq I(a|c')$.

$$I_B(a \mid c) \subseteq \tau \cap \bigcap_{(c \leq c') \in B} I_B(a \mid c') \cap \bigcap_{(c.a \subseteq d) \in B} I_B(d)$$

[13] Rappelons que la syntaxe autorise plusieurs valeurs pour l'attribut d'un objet, contrairement à la sémantique.

$$I_B(a)(I_B(o)) \in \bigcap_{(o \in c) \in B} I_B(a \mid c) \cap \bigcap_{(o.a=v) \in B} \{I_B(v)\}$$

La notion de conséquence – ce qui est vrai dans tous les modèles d'une base particulière – et de base inconsistante – base n'ayant pas de modèle – sont définies de manière classique.

DÉFINITION 7 : CONSÉQUENCE

Soient B une base et δ une assertion, δ est une conséquence de B, noté $B \models \delta$, ssi :

$$\forall \text{M modèle de } B, \models_M \delta.$$

Dans la base prise comme exemple, on a ainsi : $B \models i'' \in c_2$. Tous les modèles, en effet, satisfont $I_B(i'') \in I_B(c_2)$.

DÉFINITION 8 : BASE INCONSISTANTE

Une base B est inconsistante ssi il n'existe pas de modèle de B.

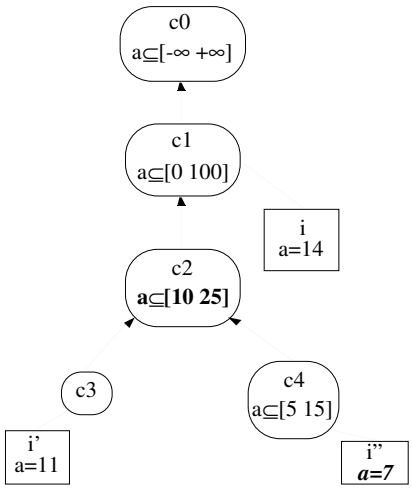


Figure 15 : exemple de base inconsistante : la valeur de l'attribut a de l'instance i'' n'appartient pas à la restriction d'attribut de la classe c_2, alors que l'instance est attachée à une sous-classe de c_2.

Par exemple, la base de la Figure 12 à laquelle est simplement ajoutée l'assertion $(i''.a=7)$ est inconsistante (Figure 15). En effet, $(i''.a=7) \in B$, $(i'' \in c_4) \in B$, $(c_2.a \subseteq [10\ 25]) \in B$, $(c_4.a \subseteq [5\ 15]) \in B$ et $(c_4 \leq c_2) \in B$. Tout modèle doit donc satisfaire : $I(a \mid c_4) \subseteq [10\ 15]$ d'où $I(a)(I(i'')) \in \{7\} \cap [10\ 25] \cap [5\ 15]$, ce qui est impossible.

3. Déduction syntaxique

Calculer tous les modèles d'une base de connaissance pour en connaître les conséquences est impossible mais connaître ses conséquences est nécessaire pour l'exploiter. C'est pourquoi, un système déductif est défini par un ensemble de règles d'inférence. Il doit

permettre de déduire toutes les conséquences de la base et seulement elles. La complétude avec la sémantique est étudiée au paragraphe b.

a) Système déductif

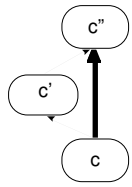
$\frac{\delta_1 \ldots \delta_n}{\delta}$ *co* signifie que si les assertions $\delta_1, \ldots \delta_n$ peuvent être déduites de la base et que la condition *co* est vérifiée alors δ est également déductible de la base. La condition *co* est distinguée des assertions car les conditions portent sur les types et ne sont pas des assertions de la base. Les règles sont illustrées par des figures dans lesquelles l'assertion déduite est indiquée en gras.

DÉFINITION 9 : RÈGLES D'INFÉRENCE

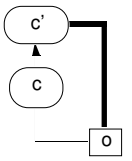
Pour tout $c, c', c'', c_i \in C$, $o \in O$, $a \in A$, $v \in V$ et $d, d', d_i \in T$:

(1) (≤-réflexivité) $\quad \dfrac{}{c \leq c}$

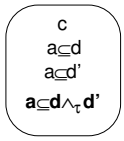
(2) (≤-transitivité) $\quad \dfrac{c \leq c' \quad c' \leq c''}{c \leq c''}$



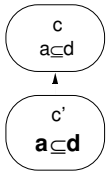
(3) (∈-clôture) $\quad \dfrac{c \leq c' \quad o \in c}{o \in c'}$



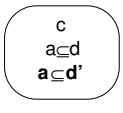
(4) (d-intersection) $\quad \dfrac{c.a \subseteq d \quad c.a \subseteq d'}{c.a \subseteq d \wedge_\tau d'}$



(5) (d-héritage) $\quad \dfrac{c.a \subseteq d \quad c' \leq c}{c'.a \subseteq d}$

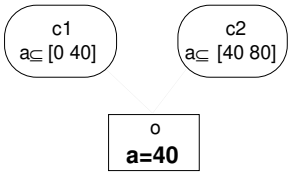


(6) (d-clôture) $\quad \dfrac{c.a \subseteq d}{c.a \subseteq d'} \quad d <:_\tau d'$



$d <:_\tau d'$

(7) (o-valuation) $\quad \dfrac{\{c_i.a \subseteq d_i \quad o \in c_i\}}{o.a = v} \quad \{v\} = \bigwedge_i d_i$



Les deux premières règles expriment la réflexivité et la transitivité de la relation de spécialisation, la troisième exprime qu'une instance appartient à toutes les super-classes de la classe à laquelle elle est attachée. Les quatrième et cinquième règles signifient que la contrainte portant sur un attribut d'une classe est l'accumulation de toutes celles posées sur cet attribut dans la classe et dans ses super-classes. La sixième règle est une règle technique permettant d'obtenir la complétude ; elle permet d'ajouter des contraintes plus faibles que la contrainte existante. Enfin, la dernière règle indique que si, pour un ensemble de classes, l'intersection des restrictions de domaine d'un attribut est réduit à un singleton, les instances qui appartiennent à l'ensemble de ces classes possèdent cette valeur pour l'attribut considéré.

EXEMPLES :

Voyons quelques exemples d'application de ces règles sur la base de la Figure 12 (page 51).

(1) (≤-réflexivité) $\dfrac{}{c_3 \leq c_3}$ (2) (≤-transitivité) $\dfrac{c_3 \leq c_2 \quad c_2 \leq c_1}{c_3 \leq c_1}$

(3) (∈-clôture) $\dfrac{c_3 \leq c_2 \quad i' \in c_3}{i' \in c_2}$ (4) (d-intersection) $\dfrac{c_4.a \subseteq [10\ 25] \quad c_4.a \subseteq [5\ 15]}{c_4.a \subseteq [10\ 15]}$

(5) (d-héritage) $\dfrac{c_2.a \subseteq [10\ 25] \quad c_4 \leq c_2}{c_4.a \subseteq [10\ 25]}$ (6) (d-clôture) $\dfrac{c_2.a \subseteq [10\ 25]}{c_2.a \subseteq [8\ 50]}\ [10\ 25] < :[8\ 50]$

VOCABULAIRE :

La relation de spécialisation étant transitive, les *classes les plus spécifiques* d'un ensemble de classes sont celles qui n'admettent pas de sous-classes dans cet ensemble.

La clôture déductive d'une base est le plus petit ensemble contenant la base et clos par déduction.

DÉFINITION 10 : CLÔTURE D'UNE BASE

Soit B une base, la clôture de la base B est la base, notée Cn(B), telle que :

(1) $B \subseteq$ Cn(B),

(2) Pour tous $\delta_1, \ldots \delta_n \in$ Cn(B) et toute règle $\dfrac{\delta_1 \ldots \delta_n}{\delta}\ co$, si co est satisfaite, $\delta \in$ Cn(B),

(3) Cn(B) ne contient pas d'autres assertions.

Il est à présent possible de déterminer quelles sont les assertions qui peuvent être déduites de la base.

DÉFINITION 11 : DÉDUCTION

Une base B permet de déduire une assertion δ, noté $B \vdash \delta$, ssi $\delta \in$ Cn(B).

Une base B permet de déduire une base B', noté $B \vdash B'$, ssi pour tout $\delta \in B'$, $B \vdash \delta$.
La négation de la déduction est notée $\nvdash$.

La base prise comme exemple (Figure 12) permet donc de déduire :

$B \vdash c_4.a \subseteq [10\ 25]$ par application du (d-héritage) grâce à $(c_4 \leq c_2) \in B$ et $(c_2.a \subseteq [10\ 25]) \in B$
$B \vdash c_4.a \subseteq [10\ 15]$ par application de la (d-intersection) grâce à $(c_4.a \subseteq [5\ 15]) \in B$ et $B \vdash c_4.a \subseteq [10\ 25]$.

Dans les systèmes à objets, la déduction correspond à l'héritage, à l'inférence de type ou de valeur. Remarquons que toute assertion appartenant à Cn(B) est déductible en utilisant un nombre fini de pas de déduction. En effet, une base contient un nombre fini n d'assertions par hypothèse et donc les assertions de la forme $(c \leq c')$ se déduisent en au plus n applications de la règle de ($\leq$-transitivité). Les assertions de la forme $(o \in c)$ se déduisent alors en *une* application de la règle de ($\in$-clôture). Pour les assertions de la forme $(c.a \subseteq d)$, il s'agit d'appliquer la règle de (d-héritage) pour obtenir les restrictions qui pèsent sur les super-classes et la règle de (d-intersection) pour obtenir la restriction résultante. Il ne reste alors qu'à appliquer la règle de (d-clôture). Pour les assertions de la forme $(o.a = v)$, la règle de (o-valuation) est appliquée sur toutes les assertions qui expriment le rattachement de l'instance à des classes et les restrictions d'attribut qui pèsent sur ces classes.

Les propositions suivantes établissent les propriétés de système déductif de $\vdash$.

PROPOSITION 2 :
La relation entre bases $\vdash$ est transitive.

PROPOSITION 3 :
Soient B et B' deux bases, $B \vdash B'$ ssi $B \vdash$ Cn(B').

Il est également possible d'exprimer à l'aide de la déduction la notation $(c \leq^* c')$.

PROPOSITION 4 :
$(c \leq^* c') \in B$ ssi $B \vdash c \leq c'$.

L'une des caractéristiques intéressante de la déduction est son coût polynomial.

PROPOSITION 5 : COMPLEXITÉ DE LA DÉDUCTION
La déduction, c'est-à-dire déterminer si $B \vdash \delta$ pour B et δ donnés, a une complexité polynomiale en nombre d'assertion de la base.

Nous disposons donc d'un système déductif pour les bases exprimées dans le langage par objets ; il reste à vérifier qu'il correspond à la sémantique posée pour ce langage, c'est-à-dire que toutes les assertions qui sont déductibles sont des conséquences et réciproquement.

b) Correction et complétude de la déduction syntaxique

Etablir le pont entre la déduction syntaxique ($\vdash$) et la conséquence sémantique ($\models$) permet de s'assurer que ce qui est déduit syntaxiquement est bien valide sémantiquement (correction), et vice versa (complétude faible).

PROPOSITION 6 : CORRECTION

Soient B une base et δ une assertion, si $B \vdash \delta$ alors $B \models \delta$.

Esquisse de démonstration : La correction est assez naturellement acquise, vu que les règles d'inférence décrivent un mécanisme de déduction conforme aux propriétés classiques d'une fonction d'interprétation. Chacune des règles se justifie facilement en utilisant les propriétés ensemblistes. Prenons juste un exemple, la règle de (d-héritage) : si $B \models c.a \subseteq d$ et $B \models c' \leq c$, alors $\forall M=<D,I>$ tel que $\models_M B$, $I(a|c) \subseteq I(d)$ et $I(a|c') \subseteq I(a|c)$ car $I(c') \subseteq I(c)$, d'où $I(a|c') \subseteq I(d)$ $\forall M$ modèle de B, $\models_M c'.a \subseteq d$ donc $B \models c'.a \subseteq d$. ◊

Etudions la réciproque : ce qui est vrai dans tous les modèles est-il déductible par les règles d'inférence ?

En logique classique, lorsqu'il y a inconsistance, il n'existe pas de modèle de la base et donc tout est vrai. Afin d'avoir complétude, si la base est inconsistante, tout doit donc être déductible par les règles d'inférence. Dans notre cadre, une base inconsistante a pour conséquence, de la même manière, n'importe quelle assertion, mais elle ne permet pas de déduire n'importe quelle assertion par les règles d'inférence que nous avons données. Pouvoir tout déduire en présence d'une inconsistance n'est pas souhaitable, car il ne serait plus possible alors de cibler les causes de l'inconsistance, ce qui est une propriété de localité typique des systèmes à objets. Donc, même s'il est possible d'obtenir la complétude en ajoutant des règles permettant de déduire n'importe quelle assertion lorsque la base est inconsistante, il est préférable de ne pas le faire afin que la révision puisse se focaliser sur les assertions qui participent à l'inconsistance. La complétude n'est donc acquise que lorsque la base est consistante, ce qui explique qu'elle est qualifiée de 'faible'.

PROPOSITION 7 : COMPLÉTUDE FAIBLE

Soient une base B consistante et une assertion δ, si $B \models \delta$ alors $B \vdash \delta$.

Esquisse de démonstration : l'étape la plus délicate consiste à déterminer pour chaque forme d'assertion conséquence d'une base ($B \models \delta$), les assertions de la base qui permettent de

la déduire. Pour ce faire, on utilise l'interprétation élémentaire – une interprétation particulière qui colle au plus près de la base, qui est un modèle si la base est consistante et sinon qui permet de déterminer les ensembles d'assertions qui provoquent l'inconsistance. On prouve le théorème en respectant un ordre sur les assertions qui permet d'utiliser les résultats intermédiaires de la démonstration (ainsi pour $(o \in c)$, le résultat sur $(c' \leq c)$ est utilisé) : pour chacune, on dispose donc des ensembles d'assertions qui permettent de les déduire et le résultat est prouvé par l'application des règles du système déductif. ◊

La définition de cette interprétation particulière est donnée en annexe, de même que la preuve des théorèmes, pour le langage étendu défini en partie B.

4. Forme première

Les systèmes de représentation de connaissance ne manipulent pas les bases sous forme de modèle ou de clôture déductive car la clôture déductive d'une base est souvent infinie, même dans le cas simple présenté ici à cause des intervalles d'entiers[14]. Les bases sont manipulées sous une forme normalisée permettant de déterminer rapidement inconsistance et conséquence.

Lors de la révision, il est nécessaire de supprimer des assertions pour que l'ajout de la connaissance problématique préserve la consistance ; ces suppressions doivent être effectives, c'est-à-dire ne plus pouvoir être déduites de la base, sous peine d'obtenir une base inconsistante. Or, si des assertions sont supprimées dans une base quelconque, il est possible qu'elles soient encore déductibles de la base, ce qui complique fortement le processus de révision car il faut faire en sorte que l'assertion à supprimer ne soit plus déductible. L'un des exemples les plus simples est le suivant : soit la base $B=\{o\in c,\ o\in c',\ c\leq c'\}$ dans laquelle $(o\in c')$ doit être supprimée ; cette assertion est encore déductible de $B'=\{o\in c,\ c\leq c'\}$.

a) Définition

La forme première d'une base est le sous-ensemble de la clôture déductive de la base qui répond aux contraintes suivantes :

- la base initiale peut en être déduite et donc la forme première peut être utilisée à sa place ;

---

[14] Dans notre cas, il est possible de définir la clôture déductive de telle sorte qu'elle ne soit pas infinie. Il suffit de remplacer la règle (6) par la règle (6') ci-dessous proposée par Roland Ducournau.

(6') : (d-clôture) $\dfrac{c'.a \subseteq d'\ \ c.a \subseteq d}{c.a \subseteq d'}$ $\qquad d <:_\tau d'$

Ceci permet de limiter la déduction de nouveaux domaines aux domaines qui sont définis pour une autre classe et donc de n'avoir qu'un nombre fini de telles assertions par restriction d'attribut de classe.

- les suppressions y sont effectives : si une assertion est supprimée d'une base sous forme première, celle-ci ne peut plus être déduite.

Un tel ensemble est fini (car la base initiale l'est) et minimal au sens de l'inclusion ensembliste (à cause de la deuxième contrainte).

DÉFINITION 12 : FORME PREMIÈRE

Une forme première d'une base B, notée FP(B), est telle que :

(1) $B \vdash \mathrm{FP}(B)$,

(2) $\mathrm{FP}(B) \vdash B$,

(3) pour toute assertion δ appartenant à FP(B), $\mathrm{FP}(B)-\{\delta\} \nvdash \delta$.

La définition ne permettant pas de calculer la forme première, nous en proposons une caractérisation algébrique.

Il est intéressant de définir une base intermédiaire entre la clôture déductive et la forme première, appelée *forme normale* qui contient les assertions qui décrivent de manière minimale les classes et les instances :

- chaque classe spécialise ses super-classes les plus spécifiques,
- chaque attribut de classe est contraint par le domaine le plus restreint,
- chaque instance est attachée à ses classes d'appartenance les plus spécifiques,
- les attributs des instances sont valués par leur valeur éventuelle dans la clôture déductive.

Pour obtenir la forme normale, des retraits sont effectués dans la clôture déductive afin de ne garder que les assertions nécessaires ; ces retraits sont le pendant des règles de déduction ($\leq$-transitivité), ($\in$-clôture) et (d-clôture). L'utilisation de la clôture déductive n'est pas nécessaire pour le calcul de la forme normale et de la forme première, comme le montre l'algorithme présenté au chapitre IV, mais cela rend la définition plus compréhensible, notamment pour les restrictions des attributs de classe.

DÉFINITION 13 : FORME NORMALE

Soient une base B et $\mathrm{Cn}'(B)=\mathrm{Cn}(B)-\{(c\leq c)\in \mathrm{Cn}(B)\}$, la réduction réflexive de Cn(B). La forme normale d'une base B, notée FN(B), se définit par :

$\mathrm{FN}(B) = \mathrm{Cn}'(B)$

$- \{(c \leq c'') \ ; \ (c \leq c'), (c' \leq c'') \in \mathrm{Cn}'(B)\}$ ($\leq$-réduction)

$- \{(o \in c) \ ; \ (o \in c'), (c' \leq c) \in \mathrm{Cn}'(B)\}$ ($\in$-réduction)

$- \{(c.a \subseteq d) \ ; \ (c.a \subseteq d') \in \mathrm{Cn}'(B) \text{ et } d' <:_\tau d\}$ (d-normalisation)

EXEMPLE :

Voici la forme normale de la base prise comme exemple jusqu'à présent :

$B=\{c_1 \leq c_0 ; c_2 \leq c_1 ; c_3 \leq c_2 ; c_4 \leq c_2 ;$	$\mathrm{FN}(B)=\{c_1 \leq c_0 ; c_2 \leq c_1 ; c_3 \leq c_2 ; c_4 \leq c_2 ;$
$i \in c_2 ;\ i' \in c_3 ;\ i'' \in c_4 ;\ c_0.a \subseteq [-\infty\ +\infty] ;$	$i \in c_2 ;\ i' \in c_3 ;\ i'' \in c_4 ;\ c_0.a \subseteq [-\infty\ +\infty] ;$
$c_1.a \subseteq [0\ 100] ; c_2.a \subseteq [10\ 25] ;$	$c_1.a \subseteq [0\ 100] ; c_2.a \subseteq [10\ 25] ;$
$c_4.a \subseteq [5\ 15] ;$	$\boldsymbol{c_3.a \subseteq [10\ 25] ;\ c_4.a \subseteq [10\ 15] ;}$
$i.a=14 ; i'.a=11\}$	$i.a=14 ; i'.a=11\}.$

Il est nécessaire de considérer l'ensemble $\mathrm{Cn}'(B)$, car la réduction réflexive doit être réalisée avant les autres pour ne faire que les retraits pertinents. Si tel n'est pas le cas, toutes les assertions de la forme ($c \leq c'$), par exemple, seraient supprimées par l'opération de (≤-réduction).

La forme première permet de ramener une base de connaissance à une forme unique qui contient le minimum d'assertions nécessaires pour déduire la base. Elle supprime de la clôture déductive *toutes* les assertions qui sont déductibles par les règles d'inférence. Elle correspond à la forme normale de laquelle on a supprimé :

- les valeurs d'attribut d'instance qui sont déductibles par la règle de (o-valuation),
- les restrictions de domaines qui sont présentes dans l'une des super-classes ou calculable par intersection des domaines des super-classes.

PROPOSITION 8 : CARACTÉRISATION DE LA FORME PREMIÈRE

Soit une base B. La forme première de B se calcule par :

$\mathrm{FP}(B) = \mathrm{FN}(B)$

$$- \{(o.a = v)\ ;\ \{c_i.a \subseteq d_i \quad o \in c_i\} \in \mathrm{Cn}'(B) \text{ et } \{v\} = \bigwedge_i d_i\} \qquad \text{(o-anti-valuation)}$$

$$- \{(c.a \subseteq d)\ ;\ \{c_i.a \subseteq d_i \quad c \leq c_i\} \in \mathrm{Cn}'(B),\ \bigwedge_i d_i = d\} \qquad \text{(d-anti-intersection)}$$

Esquisse de démonstration : Il faut démontrer d'une part que la base ainsi calculée permet de déduire la base initiale et réciproquement (1) et que le retrait d'assertion y est définitif (2).

(1) il est immédiat que la base permet de déduire la forme première ; pour montrer que la base obtenue par le calcul ci-dessus permet de déduire la base, chaque type d'assertion est étudiée : si l'assertion est supprimée de la clôture déductive, on peut en déduire les ensembles d'assertions nécessaires qui appartiennent à la clôture déductive, puis les ensembles d'assertions qui appartiennent à la forme première. Il ne reste alors qu'à appliquer les règles du système déductif. De plus, les assertions sont traitées dans un ordre qui permet d'utiliser les premiers résultats de la démonstration.

(2) la preuve se fait par l'absurde en supposant que l'assertion δ est déductible de $\mathrm{FP}(B)-\{\delta\}$. Or, si elle est déductible et qu'elle n'appartient pas à $\mathrm{FP}(B)-\{\delta\}$, on peut en déduire les assertions qui doivent appartenir à $\mathrm{FP}(B)-\{\delta\}$ en fonction des règles d'inférence qui

permettent de la déduire. Ceci permet d'aboutir à des ensembles d'assertions qui appartiennent à FP(B)–{δ}, ce qui, conjugué aux retraits pour obtenir la forme première, montre que δ n'appartient pas à la forme première. ◊

b) Propriétés

PROPOSITION 9 : UNICITÉ DE LA FORME PREMIÈRE

Soit une base B ; il existe une unique forme première FP(B).

La forme première ne contient que les assertions qui indiquent :

- l'attachement d'une instance à ses classes d'appartenance les plus spécifiques,
- la relation de spécialisation d'une classe avec ses super-classes les plus spécifiques,
- la valeur d'un attribut d'une instance si celle-ci n'est pas déductible des classes auxquelles appartient l'instance,
- la restriction de domaine d'un attribut de classe si cette restriction n'est ni présente dans l'une de ses super-classes ni calculable par intersection des domaines des super-classes (Figure 17) ; le second cas ne se distingue du premier que dans le cas de la multi-spécialisation, car avec l'hypothèse de mono-spécialisation la restriction d'attribut de la super-classe est l'intersection des restrictions posées sur toutes les super-classes.

Justifions les deux types de suppressions effectuées sur la forme normale pour obtenir la forme première.

La (o-anti-valuation) consiste à supprimer les valeurs d'attribut d'instance qui sont déductibles par la règle de (o-valuation), car les suppressions doivent être effectives dans la forme première. Si $(o.a = v)$ appartient à la base et qu'un l'ensemble d'assertions $\{c_i.a \subseteq d_i \quad o \in c_i\}$ est déductible de la forme première avec $\{v\} = \wedge_i d_i$, même si ($o.a$=v) n'appartient pas à la forme première, ($o.a$=v) est déductible de la forme première. La Figure 16 illustre la règle de (o-anti-valuation).

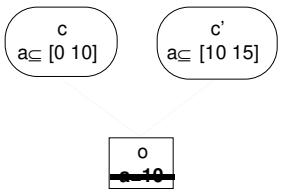


Figure 16 : B= { $o \in c$; $o \in c'$; $c.a \subseteq [0\ 10]$; $c'.a \subseteq [10\ 15]$; o=10} ; sa forme première est :

FP(B)= { $o \in c$; $o \in c'$; $c.a \subseteq [0\ 10]$; $c'a \subseteq [10\ 15]$} par application de la (o-anti-valuation).

De plus, les règles de (d-intersection) et de (d-héritage) ne trouvent pas leur pendant direct dans cette définition mais sont regroupées dans la (d-anti-intersection). Si on appliquait les

suppressions correspondant à la règle de (d-intersection), trop d'assertions sur les restrictions d'attribut seraient perdues pour pouvoir redéduire la base. Cela est montré dans l'exemple qui suit.

EXEMPLE : SUPPRESSIONS PAR (D-ANTI-INTERSECTION)

Soit la base : $B=\{c\leq c',\ c'.a\subseteq[0\ 20],\ c.a\subseteq[10\ 30]\}$

$\mathrm{FN}(B)=\{c\leq c',\ c'.a\subseteq[0\ 20],\ c.a\subseteq[10\ 20]\}$

Comme $(c.a\subseteq[10\ 30])\in \mathrm{Cn}'(B)$, $(c.a\subseteq[0\ 20])\in \mathrm{Cn}'(B)$ et que $[10\ 20]=[10\ 30]\wedge[0\ 20]$, en faisant les suppressions correspondant à l'inverse de la règle de (d-intersection), $c.a\subseteq[10\ 20]$ serait supprimée de la forme première. La base obtenue est alors $\{c\leq c',\ c'.a\subseteq[0\ 20]\}$ de laquelle il n'est plus possible de déduire que $c.a\subseteq[10\ 30]$. Or, par la définition donnée, $\mathrm{FP}(B)=\{c\leq c',\ c'.a\subseteq[0\ 20],\ c.a\subseteq[10\ 20]\}$, ce qui correspond à ce qui est attendu, car la base initiale peut être redéduite et la restriction d'attribut de c est la plus restreinte possible.

Une caractérisation de la forme première qui n'utilise pas la clôture déductive est la suivante :

$\mathrm{FP}(B) = \mathrm{FN}(B)$

$-\ \{(o.a = v)\ ;\ \{c_i.a \subseteq d_i \quad o \in c_i\} \in \mathbf{FN}(\boldsymbol{B})$ et $\{v\} = \underset{i}{\wedge} d_i\}$ (o-anti-valuation)

$-\ \{(c.a \subseteq d)\ ;\ \{c_i.a \subseteq d_i \quad c \leq c_i\} \in \mathbf{FN}(\boldsymbol{B}),\ \underset{i}{\wedge} d_i = d\}$ (d-anti-intersection)

EXEMPLES : FORME PREMIÈRE

(1) La forme première de la base prise comme exemple jusqu'à présent est :

$B=\{c_1\leq c_0\ ;\ c_2\leq c_1\ ;\ c_3\leq c_2\ ;\ c_4\leq c_2\ ;$
$i\in c_2\ ;\ i'\in c_3\ ;\ i''\in c_4\ ;\ c_0.a\subseteq[-\infty\ \ +\infty]\ ;$
$c_1.a\subseteq[0\ 100]\ ;c_2.a\subseteq[10\ 25]\ ;$
$c_4.a\subseteq[5\ 15]\ ;\ i.a=14\ ;\ i'.a=11\}$

$\mathrm{FP}(B)=\{c_1\leq c_0\ ;\ c_2\leq c_1\ ;\ c_3\leq c_2\ ;\ c_4\leq c_2\ ;$
$i\in c_2\ ;\ i'\in c_3\ ;\ i''\in c_4\ ;\ c_0.a\subseteq[-\infty\ \ +\infty]\ ;$
$c_1.a\subseteq[0\ 100]\ ;\ c_2.a\subseteq[10\ 25]\ ;$
$\boldsymbol{c_4.a\subseteq[10\ \ 15]\ ;}\ i.a=14\ ;\ i'.a=11\}.$

(2)

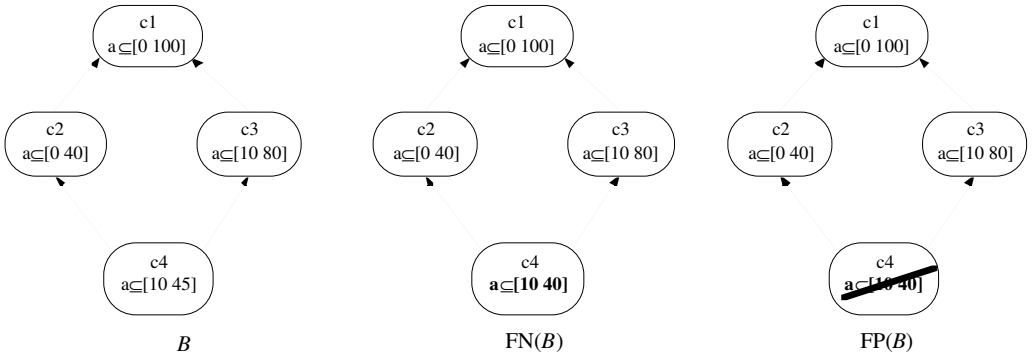


Figure 17 : $B=\ \{c_2\leq c_1\ ;\ c_3\leq c_1\ ;\ c_4\leq c_2\ ;\ c_4\leq c_3\ ;\ c_1.a\subseteq[0\ 100]\ ;\ c_2.a\subseteq[0\ 40]\ ;\ c_3.a\subseteq[10\ 80]\ ;\ c_4.a\subseteq[10\ 45]\}\ ;$
$\mathrm{FN}(B)=\ \{c_2\leq c_1\ ;\ c_3\leq c_1\ ;\ c_4\leq c_2\ ;\ c_4\leq c_3\ ;\ c_1.a\subseteq[0\ 100]\ ;\ c_2.a\subseteq[0\ 40]\ ;\ c_3.a\subseteq[10\ 80]\ ;\boldsymbol{c_4.a\subseteq[10\ 40]}\}$ et

FP(B)= $\{c_2 \leq c_1 ; c_3 \leq c_1 ; c_4 \leq c_2 ; c_4 \leq c_3 ; c_1.a \subseteq [0\ 100] ; c_2.a \subseteq [0\ 40] ; c_3.a \subseteq [10\ 80]\}$ par application de la règle de (d-anti-intersection).

La Figure 17 donne un second exemple qui illustre la différence entre la forme normale et la forme première.

PROPOSITION 10 :

La taille maximale en nombre d'assertions de la forme première est de l'ordre de :

$(|O| + |C|) * (|A| + |C|)$ dans le cas général

$(|O| * |A|) + |C| * (|A| + |C|)$ sans multi-instanciation

$|O| * (|A| + |C|) + (|A| * |C|)$ sans multi-spécialisation

$(|O| + |C|) * |A|$ sans aucun des deux

D'autres propriétés intéressantes de la forme première, justifiant son utilisation, sont données ci-dessous, notamment l'équivalence de l'égalité des clôtures déductives de deux bases et de l'égalité de leur forme première (Proposition 13). Des propositions similaires faisant intervenir la forme normale existent, mais elles sont d'intérêt moindre pour la suite de ce travail.

PROPOSITION 11 :

Cn(FP(B))=Cn(B)

FP(Cn(B))=FP(B)

FP(FP(B)) = FP(B) (stabilité)

PROPOSITION 12 :

$\forall B, B'$, les conditions suivantes sont équivalentes :

(1) FP(B')$\vdash$FP(B)

(2) $B' \vdash B$

(3) Cn(B)$\subseteq$Cn(B')

PROPOSITION 13 :

$\forall B, B'$, les trois conditions suivantes sont équivalentes :

(1) FP(B) = FP(B')

(2) $B \vdash B'$ et $B' \vdash B$

(3) Cn(B) = Cn(B')

c) Calcul

L'intérêt plus pratique de la forme première est la possibilité de l'obtenir facilement par son calcul à partir d'une base quelconque et de la manipuler de façon efficace pour déterminer si la base est consistante ou si une assertion est conséquence de la base. Pour le calcul de la forme

première, il faut remarquer que les retraits peuvent être faits séquentiellement, en particulier pour les assertions concernant les restrictions de domaine d'attribut ; en effet, commencer par normaliser réduit ces assertions à une par classe ; la (d-anti-intersection) peut alors s'appliquer sur ce nombre restreint d'assertions afin de supprimer les dernières assertions redondantes.

De plus, la forme première permet de faciliter la déduction, comme cela est montré dans l'étude de la complexité des algorithmes (annexe C). Pour déduire une assertion particulière ou vérifier qu'une assertion particulière est déductible, les procédures de déduction peuvent être guidées de manière à aboutir rapidement à une conclusion car il suffit de déclencher :

les règles qui sont utiles pour la forme de l'assertion à déduire ; ainsi, pour déduire une assertion de la forme $(c \leq c')$, nul besoin des règles de (d-héritage) ou de ($\in$-clôture)

sur des assertions dont les composants (instances ou classes) sont utiles en fonction de leur place respective par rapport à l'ordre de spécialisation entre les classes ; par exemple, pour déduire une assertion de la forme $(o \in c)$, seules les assertions portant sur les classes supérieures à c (par rapport à $\leq$) peuvent être utiles dans les règles d'inférence.

Ces algorithmes sont donnés en annexe lors de l'évaluation de la complexité (annexe C).

PROPOSITION 14 : COMPLEXITÉ DU PROBLÈME

Le calcul de la forme première a une complexité polynomiale en nombre d'assertions de la base.

B. Langage avec référence aux instances

Afin que les bases de connaissance aient un pouvoir d'expression suffisant pour décrire des domaines complexes, il est nécessaire de pouvoir exprimer des relations entre les instances. La section I.C a montré que dans TROPES, ces relations entre instances se définissent grâce aux attributs qui ne sont plus valués par une valeur de type, mais par une autre instance. Il existe un lien équivalent au niveau des classes : un attribut n'est plus retreint par un domaine de type, mais par une autre classe. Cette partie décrit la syntaxe et la sémantique d'un langage d'objets autorisant les liens entre instances par le biais des attributs.

Vu que nombre de définitions et de propriétés sont identiques à celles données dans la partie précédente, elles ne seront pas reprises ; les modifications nécessaires dans les autres seront indiquées en gras. La principale différence se situe au niveau des règles de déduction. Nous rappelons que les démonstrations des propositions se trouvent en annexe.

1. Syntaxe

a) Grammaire du langage

Le langage permet de décrire des classes en relation de spécialisation, des instances de ces classes et des attributs qui sont typés dans les classes par un domaine ou une classe et qui sont valués dans les instances par une valeur ou une instance. Les notions de consistance et de cohérence étudiées dans la section III.A permettront de définir les bases correctes.

La Figure 18 est la représentation graphique d'une base, contenant deux hiérarchies de classes : la première de six classes (c_0, c_1, c_2, c_3, c_4, c_5), deux instances (o_1 et o_2), rattachées respectivement à c_4 et c_5 et deux attribut a (entier) et b référençant la seconde hiérarchie ; la seconde hiérarchie de trois classes (c'_0, c'_1, c'_2), deux instances (o_3 et o_4) et un attribut a'.

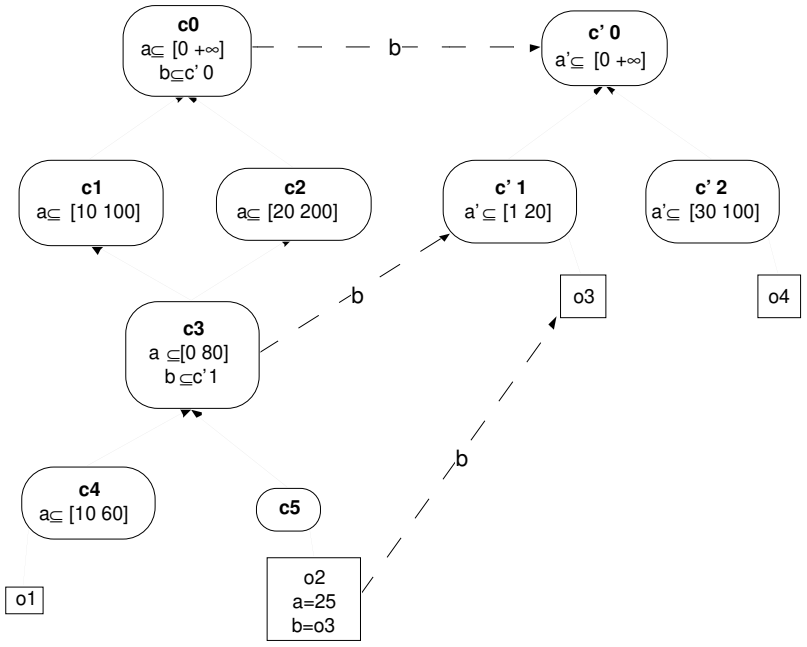


Figure 18 : Exemple de base de connaissance représentée graphiquement.

La seule modification de la grammaire se situe dans la définition de la restriction d'un attribut de classe – qui peut être à présent une autre classe – et de la valeur d'un attribut d'une instance – qui peut être une autre instance.

DÉFINITION 14 : GRAMMAIRE

<base_de_connaissance> : := {<assertion>*}

<assertion> : := <relation_spécialisation> | <rattachement_instance> | <restriction_attribut_classe> | <valeur_attribut_instance>

<relation_spécialisation> : := <nom_classe> ≤ <nom_classe>

<rattachement_instance> : := <nom_instance> ∈ <nom_classe>
<restriction_attribut_classe> : := <nom_classe>.<nom_attribut> ⊆ <domaine>
| **<nom_classe>.<nom_attribut> ⊆ <nom_classe>**
<valeur_attribut_instance> : := <nom_instance>.<nom_attribut> = <valeur>
| **<nom_instance>.<nom_attribut> = <nom_instance>**
<nom_classe> | <nom_instance> | <nom_attribut> : := <identificateur>
<domaine> et <valeur> sont définis comme précédemment.

Les contraintes méta-syntaxiques posées sur le langage simple sont valables dans ce cas également, sauf que la valeur d'attribut d'une instance peut être une autre instance. Ce langage introduit également une nouvelle relation entre classes et entre instances, la relation de référence : si ($c.a \subseteq c'$), on dira que la classe c référence la classe c' par l'attribut a. De manière similaire, si ($o.a=o'$), l'instance o référence l'instance o' par l'attribut a. Le graphe de cette relation de référence doit être sans circuit pour l'ensemble des <nom_classe> et des <nom_instance>, ce qui est détaillé dans la Définition 15.

DÉFINITION 15 : RÉFÉRENCE SANS CIRCUIT

La relation de référence dans une base est sans circuit s'il n'existe pas un ensemble d'assertions $\{[c_i.a_i \subseteq c'_i$ ou $(o_i \in c_i,\ o_i.a_i=o'_i$ et $o'_i \in c'_i)]$ et $[c'_i \leq^* c_{i+1}$ ou $c_{i+1} \leq^* c'_i]\}$ pour $i \geq 1$ tel que $c_1=c'_n$ avec $n \geq 1$.

Selon la grammaire donnée, la base de connaissance de la Figure 18 s'exprime syntaxiquement par :

$B=\{c_1 \leq c_0\ ;\ c_2 \leq c_0\ ;\ c_3 \leq c_2\ ;\ c_3 \leq c_1\ ;\ c_4 \leq c_3\ ;\ c_5 \leq c_3\ ;\ o_1 \in c_4\ ;\ o_2 \in c_5\ ;\ c_0.a \subseteq [0\ +\infty]\ ;\ c_0.b \subseteq c'_0\ ;\ c_1.a \subseteq [0\ 100]\ ;\ c_2.a \subseteq [20\ 200]\ ;\ c_3.a \subseteq [0\ 80]\ ;\ c_3.b \subseteq c'_1\ ;\ c_4.a \subseteq [10\ 60]\ ;\ o_2.a=25\ ;\ o_2.b=o_3\ ;\ c'_1 \leq c'_0\ ;\ c'_2 \leq c'_0\ ;\ o_3 \in c'_1\ ;\ o_4 \in c'_2\ ;\ c'_0.a' \subseteq [0\ +\infty]\ ;\ c'_1.a' \subseteq [1\ 20]\ ;\ c'_2.a' \subseteq [30\ 100]\}$.

Les types et leurs opérateurs sont définis et interprétés de la même manière que précédemment.

b) Notations

Les assertions sont de l'une des six formes $c \leq c'$, $o \in c$, $c.a \subseteq d$, $c.a \subseteq c'$, $o.a=v$ ou $o.a=o'$. Nous avons choisi de différencier les restrictions d'attributs de classe par des domaines ou par des classes, car les connaissances contenues dans la base ($c' \leq c''$) et les connaissances sur les types ($d <:_\tau d'$) n'ont pas le même statut. Cependant, un raisonnement similaire sera souvent suivi sur ces deux formes d'assertions très proches, de même que sur les deux assertions concernant la valeur d'un attribut d'instance qui peut être une valeur ou une autre instance.

Les ensembles d'entités sont définis de la même manière, ce qui implique cependant une prise en compte de plus de forme d'assertions. Ainsi, C_B est à présent l'ensemble des classes c introduites dans la base B par une assertion de la forme $c \leq c'$, $c' \leq c$, $c.a \subseteq c'$, $c'.a \subseteq c$, $c.a \subseteq d$ ou $o \in c$.

2. Sémantique

La sémantique est définie par une fonction d'interprétation des instances vers un domaine en tenant compte des types utilisés dans la syntaxe.

L'interprétation se définit comme pour le langage simple, sauf pour les attributs, car la valeur d'attribut d'une instance peut être une autre instance, ce qui est l'ajout par rapport à la partie précédente. Le co-domaine de l'interprétation d'un attribut est donc désormais l'union de l'ensemble des valeurs de type (τ) et de l'ensemble des instances du domaine (D).

DÉFINITION 16 : INTERPRÉTATION DES ENTITÉS D'UNE BASE

Soit une base de connaissance B et un domaine D quelconque disjoint de τ. Une interprétation est un couple $<D,I_B>$, I_B étant appelée fonction d'interprétation :

I_B :	$O_B \rightarrow D$	injective sur O_B (hypothèse de nom unique pour les objets)
	$C_B \rightarrow 2^D$	
	$A_B \rightarrow (D \rightarrow \tau \cup \boldsymbol{D})$	$I_B(a)$ est une fonction totale
	$V_B \rightarrow \tau$	
	$T_B \rightarrow 2^\tau$	

telle que I_B restreinte à V_B et T_B est confondue avec I.

EXEMPLE :

Donnons un exemple d'interprétation de l'exemple de base de la Figure 18.

Soit le domaine D={pays, ville}. L'attribut a est la population en millions d'habitants, l'attribut b la capitale du pays, l'attribut a' la population en centaine de milliers d'habitants. Les classes c_i sont interprétées par des ensembles de pays et les classes c'_i par des ensembles de villes. Les instances o_1 et o_2 sont des pays et o_3 et o_4 sont des villes.

D={pays, ville}

$I_B : C_B \rightarrow 2^D$

$c \rightarrow$ {ville, pays}

$I_B : A_B \rightarrow (D \rightarrow \tau)$

$a \rightarrow (D \rightarrow$entiers)

pays ou ville$\rightarrow$population en millions d'habitants

$a' \rightarrow (D \rightarrow$entiers)

pays ou ville→population en milliers d'habitants

$b \rightarrow (D \rightarrow D)$

pays ou ville→pays ou ville

$I_B : O_B \rightarrow D$

$o_1 \rightarrow$ France

$o_2 \rightarrow$ Pérou

$o_3 \rightarrow$ Lima

$o_4 \rightarrow$ Paris

La définition de la satisfiabilité d'une assertion est étendue pour tenir compte des deux nouvelles formes d'assertions ; cependant, cette extension est nécessaire uniquement car il a été décidé de séparer les restrictions d'attribut de classe par des domaines et par des classes et de même au niveau des valeurs d'attribut d'instance.

DÉFINITION 17 : SATISFACTION D'UNE ASSERTION

Une interprétation $<D,I>$ satisfait une assertion δ est noté $\models_{<D,I>}\delta$ et

$\models_{<D,I>} c \leq c'$	ssi	$I(c) \subseteq I(c')$
$\models_{<D,I>} c.a \subseteq d$	ssi	$I(a \mid c) \subseteq I(d)$
$\models_{<D,I>} o \in c$	ssi	$I(o) \in I(c)$
$\models_{<D,I>} o.a = v$	ssi	$I(a)(I(o)) = I(v)$
$\models_{<D,I>} c.a \subseteq c'$	ssi	$I(a\|c) \subseteq I(c')$
$\models_{<D,I>} o.a = o'$	ssi	$I(a)(I(o)) = I(o')$

EXEMPLE :

Dans l'exemple précédent, $\models_{<D,I>} o_2.b = o_3$ si et seulement si $I(b)(I(o_2)) = I(o_3)$, ce qui signifie : la capitale du Pérou est Lima.

La définition de modèle n'est pas modifiée. La seule contrainte modifiée sur les fonctions d'interprétation des modèles est celle qui porte sur les restrictions d'attributs de classe :

PROPOSITION 15 :

Si $M=<D,I_B>$ est un modèle de B, alors :

$$I_B[a \mid c] \subseteq (\tau \cup \mathbf{D}) \cap \bigcap_{(c \leq c') \in B} I_B[a \mid c'] \cap \bigcap_{(c.a \subseteq d) \in B} I_B[d]$$

Les notions de conséquence et de base inconsistante sont identiques. Cependant, la notion d'inconsistance recouvre à présent des cas plus variés, comme cela sera explicité dans la section III.A.

Donnons quelques notations permettant de décrire les références en chaîne des classes ou des instances.

NOTATIONS :

Soient B une base, $o \in O$, $a_1, a_2, \ldots, a_n \in A$, $v \in V \cup O$, $(o.a_1.a_2.\ldots.a_n =^* v) \in B$ ssi

pour $n=1$, $(o.a_1 = v) \in B$

pour $n>1$, il existe $o' \in O$ tel que $(o.a_1.a_2.\ldots.a_{n-1} =^* o') \in B$ et $(o'.a_n = v) \in B$

Soient B une base, $c \in C$, $n \geq 1$, $a_1, a_2, \ldots, a_n \in A$, $d \in T \cup C$, $(c.a_1.a_2.\ldots.a_n \subseteq^* d) \in B$ ssi

pour $n=1$, il existe $c' \in C$ tel que $(c \leq c') \in B$ et $(c'.a_1 \subseteq d) \in B$

pour $n>1$, il existe $c' \in C$ tel que $(c.a_1.a_2.\ldots.a_{n-1} \subseteq^* c') \in B$ $(c'.a_n \subseteq^* d) \in B$[15]

Remarquons que l'expression $\wedge_{c.a_1.\ldots.a_n \subseteq^* d} d$ est alors l'intersection de domaines qui prend en compte tous les chemins possibles partant de la classe c et suivant les attributs $a_1, a_2, \ldots, a_n$, Cela est illustré dans la Figure 19, exemple dans lequel $\wedge_{c.a_1.a_2.a_3 \subseteq^* d} d = d_1 \wedge d_2 \wedge d_3$.

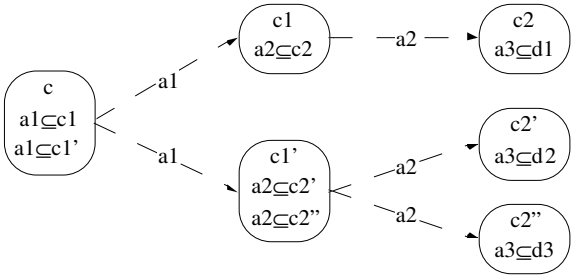


Figure 19 : Illustration de la notation sur la référence en chaîne des attributs : $(c.a_1.a_2.a_3 \subseteq^* d_1) \in B$, $(c.a_1.a_2.a_3 \subseteq^* d_2) \in B$ et $(c.a_1.a_2.a_3 \subseteq^* d_3) \in B$.

3. Déduction syntaxique

Les règles d'inférence précédentes (Définition 9, page 61) sont reprises et celles correspondant aux références à des instances sont ajoutées. Certaines sont le pendant des règles qui agissaient sur les restrictions d'attributs par des domaines ; d'autres sont nécessaires pour qu'il y ait complétude avec la sémantique. Les nouvelles règles sont illustrées par des figures dans lesquelles l'assertion déduite est indiquée en gras.

[15] il a fallu introduire * pour le cas n=1, car $c.a \subseteq^* d$ et $c.a \subseteq d$ ne sont pas identiques ; la notation $=^*$ est introduite par similitude.

DÉFINITION 18 : RÈGLES D'INFÉRENCE

Pour tout $c, c', c'', c_i \in C$, $o \in O$, $a \in A$, $v \in V$ et $d, d', d_i \in T$:

(1) (≤-réflexivité) $$\frac{}{c \leq c}$$

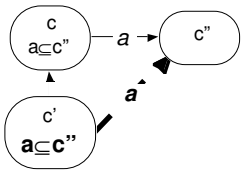
(2) (≤-transitivité) $$\frac{c \leq c' \quad c' \leq c''}{c \leq c''}$$

(3) (∈-clôture) $$\frac{c \leq c' \quad o \in c}{o \in c'}$$

(4) (d-intersection) $$\frac{c.a \subseteq d \quad c.a \subseteq d'}{c.a \subseteq d \wedge_\tau d'}$$

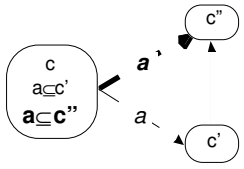
(5) (d-héritage) $$\frac{c.a \subseteq d \quad c' \leq c}{c'.a \subseteq d}$$

(5') (c-héritage) $$\frac{c.a \subseteq c'' \quad c' \leq c}{c'.a \subseteq c''}$$



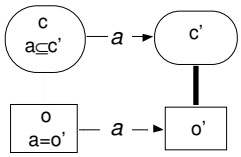
(6) (d-clôture) $$\frac{c.a \subseteq d}{c.a \subseteq d'} \quad d <:_\tau d'$$

(6') (c-clôture) $$\frac{c.a \subseteq c' \quad c' \leq c''}{c.a \subseteq c''}$$



(7) (o-valuation) $$\frac{\{c_i.a \subseteq d_i \quad o \in c_i\}}{o.a = v} \quad \{v\} = \wedge_i d_i$$

(8) (∈-inférence) $$\frac{c.a \subseteq c' \quad o \in c \quad o.a = o'}{o' \in c'}$$



Les cinquième et sixième règles sont dédoublées ; des règles similaires à celles sur les domaines sont nécessaires pour les restrictions d'attributs de classe par des classes. La huitième règle de (∈-inférence) indique qu'une instance o' référencée par une instance o d'une classe c doit appartenir à la classe c' qui est le domaine de l'attribut de la classe c. Cette règle, proposée dans [Ducournau96a], est nécessaire pour avoir la complétude avec la déduction

sémantique, car lorsque l'on considère l'ensemble des interprétations d'une base, seules celles qui permettent de déduire l'appartenance à la classe de l'instance référencée sont des modèles : cette assertion est donc déductible de la base.

Les quatrième et septième règles n'ont pas de pendant pour les restrictions par des classes. Pour la quatrième, comme l'intersection de deux classes n'est pas définie, elle n'a pas d'intérêt. Quant à la septième règle, la transposer pour des restrictions par des classes et des instances conduirait à la déduction d'assertion non satisfaite par tous les modèles. Cette règle consisterait à déduire que la valeur d'attribut d'une instance *o* est la seule instance *o'* qui appartient à l'intersection des classes *c'i* qui restreignent un attribut des classes *ci* d'appartenance de *o*. Or, une telle déduction n'est pas valide (Figure 18). En effet, il existe un modèle dont le domaine contient un élément arbitraire *e*, qui n'est l'interprétation d'aucune instance de la base et qui appartient aux interprétations de toutes les classes *c'i*, ce modèle étant tel que la valeur de l'interprétation de l'attribut de l'interprétation de *o* est *e*. Par conséquent, l'assertion *o.a=o'* n'est pas une conséquence de la base. Cette règle n'est donc pas valide car la fonction d'interprétation n'est pas surjective.

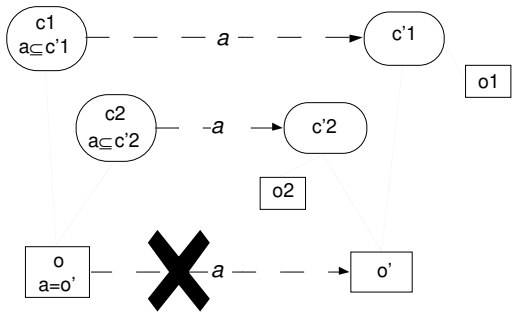


Figure 20 : une règle qui serait le pendant de la septième règle consisterait à dire que *o.a=o'* est déductible. Or, cette assertion n'est pas une conséquence sémantique de la base, car il existe une interprétation des classes *c'1* et *c'2* qui contiennent une autre instance, arbitraire.

La clôture déductive d'une base et la déduction sont définies comme précédemment et les propriétés sont conservées.

Ci-dessous est donnée une propriété attendue reliant la déduction syntaxique et les notations introduites $(c.a_1.a_2.\ldots a_n \subseteq *c') \in B$ et $(o.a_1.a_2.\ldots a_n = *o') \in B$.

PROPOSITION 16 :

S'il existe $o \in O$, a_1, $a_2,\ldots,a_n \in A$, $o' \in O$, $c' \in C$ tels que $(c.a_1.a_2.\ldots a_n \subseteq *c') \in B$, $(o.a_1.a_2.\ldots a_n = *o') \in B$, $(o \in c) \in B$ alors $B \vdash o' \in c'$.

Cette proposition est illustrée par la Figure 21.

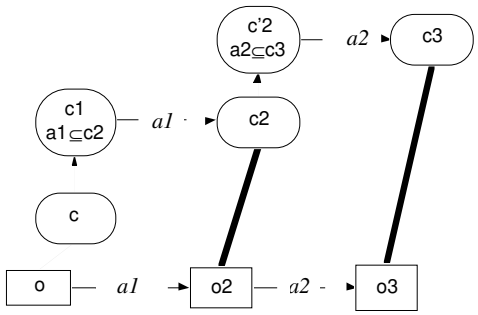


Figure 21 : $(o.a_1.a_2 = {}^*o_3) \in B$ et $(c.a_1.a_2 \subseteq {}^*c_3) \in B$ d'où l'on peut déduire $B \vdash o_3 \in c_3$.

Les propriétés de correction et de complétude faible sont vérifiées comme précédemment (paragraphe A.3.b).

complément de démonstration. Vu les modifications apportées au langage par rapport à la partie A, l'interprétation élémentaire a dû être modifiée afin de tenir compte des références aux instances et afin de toujours permettre de détecter les inconsistances. De plus, toute classe cohérente (pouvant contenir des instances) doit être interprétée par un ensemble non vide et satisfaire les relations de spécialisation et de référence, afin de détecter les classes vides, qui seront définies comme incohérentes : l'interprétation de toute classe incohérente est vide et l'interprétation de toute classe cohérente contient au moins une instance, qui est une instance arbitraire du domaine si une telle instance n'existe pas dans la base. ◊

4. Forme première

La forme première est définie comme dans le langage simple, mais la définition de la forme normale et la caratérisation de la forme première doivent être adaptées afin de tenir compte des nouvelles règles d'inférence relatives aux références aux instances.

DÉFINITION 19 : FORME NORMALE (LANGAGE COMPLEXE)

Soient une base B et $\mathrm{Cn}'(B)=\mathrm{Cn}(B) - \{c \leq c \in \mathrm{Cn}(B)\}$, la réduction réflexive de $\mathrm{Cn}(B)$. La forme normale d'une base B, notée $\mathrm{FN}(B)$, se définit par :

$\mathrm{FN}(B) = \mathrm{Cn}'(B)$

$-\{(c \leq c'') \; ; \; (c \leq c'), (c' \leq c'') \in \mathrm{Cn}'(B)\}$ (≤-réduction)

$-\{(o \in c) \; ; \; (o \in c'), (c' \leq c) \in \mathrm{Cn}'(B)\}$ (∈-réduction)

$-\{(c.a \subseteq d) \; ; \; (c.a \subseteq d') \in \mathrm{Cn}'(B) \text{ et } d' <:_\tau d\}$ (d-normalisation)

$-\{(c.a \subseteq c') \; ; \; (c.a \subseteq c''), (c'' \leq c') \in \mathrm{Cn}'(B)\}$ (c-réduction)

La forme normale contient donc les assertions indiquant :

- les liens de spécialisation d'une classe avec ses super-classes les plus spécifiques,
- le rattachement d'une instance à ses classes d'appartenance les plus spécifiques,
- le domaine normalisé pour chaque couple (attribut, classe),

- les valeurs des attributs des instances si elles existent dans la clôture déductive,
- les classes les plus spécifiques qui restreignent les attributs de classes.

PROPOSITION 17 : FORME PREMIÈRE (LANGAGE COMPLEXE)

Soit une base B. La forme première de B se calcule par :

$FP(B) = FN(B)$

$- \{(o.a = v)\ ;\ \{c_i.a \subseteq d_i \quad o \in c_i\} \in \mathrm{Cn}'(B)$ et $\{v\} = \bigwedge_i d_i\}$ (o-anti-valuation)

$- \{(c.a \subseteq d)\ ;\ \{c_i.a \subseteq d_i \quad c \leq c_i\} \in \mathrm{Cn}'(B),\ \bigwedge_i d_i = d\}$ (d-anti-intersection)

$- \{(c.a \subseteq c')\ ;\ (c''.a \subseteq c'), (c \leq c'') \in \mathbf{Cn}'(B)\}$ (c-anti-héritage)

$- \{(o \in c)\ ;\ (o' \in c'),\ (c'.a \subseteq c'), (o'.a = o) \in \mathbf{Cn}'(B)\}$ ($\in$-anti-inférence)

La forme première contient comme précédemment le minimum d'assertions permettant de redéduire la base entière, c'est-à-dire :

- les liens de spécialisation d'une classe avec ses super-classes les plus spécifiques,
- le rattachement d'une instance à ses classes les plus spécifiques, uniquement s'il n'est pas déductible par les références faites vers cette instance,
- le domaine normalisé pour chaque couple (attribut, classe), sauf s'il est déductible des restrictions posées sur les super-classes,
- les valeurs des attributs des instances si elles ne sont pas déductibles,
- les classes les plus spécifiques qui restreignent les attributs de classes, sauf si ceci est déductible des restrictions posées sur les super-classes.

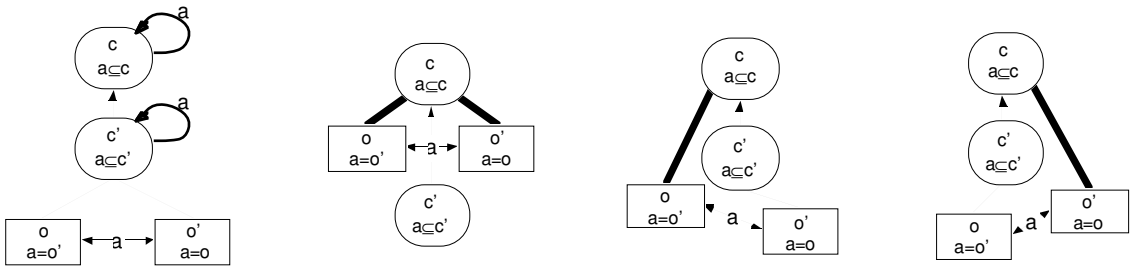
EXEMPLE :

La base prise comme exemple jusqu'à présent (Figure 18 page 72) est déjà pratiquement sous forme première :

$B = \{c_1 \leq c_0\ ;\ c_2 \leq c_0\ ;\ c_3 \leq c_2\ ;\ c_3 \leq c_1\ ;\ c_4 \leq c_3\ ;\ c_5 \leq c_3\ ;\ o_1 \in c_4\ ;\ o_2 \in c_5\ ;\ c_0.a \subseteq [-\infty\ +\infty]\ ;\ c_0.b \subseteq c'_0\ ;\ c_1.a \subseteq [0\ 100]\ ;\ c_2.a \subseteq [20\ 200]\ ;\ c_3.a \subseteq [0\ 80]\ ;\ c_3.b \subseteq c'_1\ ;\ c_4.a \subseteq [10\ 60]\ ;\ o_2.a = 25\ ;\ o_2.b = o_3\ ;\ c'_1 \leq c'_0\ ;\ c'_2 \leq c'_0\ ;\ o_3 \in c'_1\ ;\ o_4 \in c'_2\ ;\ c'_0.a' \subseteq [-\infty\ +\infty]\ ;\ c'_1.a' \subseteq [1\ 20]\ ;\ c'_2.a' \subseteq [30\ 100]\}\}$

$FP(B) = \{c_1 \leq c_0\ ;\ c_2 \leq c_0\ ;\ c_3 \leq c_2\ ;\ c_3 \leq c_1\ ;\ c_4 \leq c_3\ ;\ c_5 \leq c_3\ ;\ o_1 \in c_4\ ;\ o_2 \in c_5\ ;\ c_0.a \subseteq [-\infty\ +\infty]\ ;\ c_0.b \subseteq c'_0\ ;\ c_1.a \subseteq [0\ 100]\ ;\ c_2.a \subseteq [20\ 200]\ ;\ \mathbf{c_3.a \subseteq [20\ 80]}\ ;\ c_3.b \subseteq c'_1\ ;\ c_4.a \subseteq [10\ 60]\ ;\ o_2.a = 25\ ;\ o_2.b = o_3\ ;\ c'_1 \leq c'_0\ ;\ c'_2 \leq c'_0\ ;\ o_3 \in c'_1\ ;\ o_4 \in c'_2\ ;\ c'_0.a' \subseteq [-\infty\ +\infty]\ ;\ c'_1.a' \subseteq [1\ 20]\ ;\ c'_2.a' \subseteq [30\ 100]\}$

Les propriétés de la forme première sont toujours vérifiées, mais notons cependant que l'unicité de la forme première ne peut plus être obtenue si l'hypothèse qu'il n'y a pas de références circulaires n'est pas posée (Figure 22).



(a) - Base

(b) - Base calculée selon la définition de la forme première

(c) - Forme première - 1

(d) - Forme première - 2

Figure 22 : La base donnée (a) en appliquant la règle de ($\in$-anti-inférence) conduit à la base (b), ce qui n'est pas une forme première, dans le sens donné ici, car elle ne permet pas de déduire la base initiale. De plus, en cherchant les bases, sous-ensembles minimaux qui permettent de redéduire la base initiale, force est d'admettre qu'il en existe plusieurs (c et d), parmi lesquels il n'est pas possible de choisir de manière non arbitraire.

C. Comparaison avec d'autres approches

Nous allons comparer le langage défini dans la partie précédente avec les approches présentées dans le chapitre I qui sont particulièrement intéressante pour la suite de ce travail, soit le langage terminologique de Nebel et TROPES.

1. Langage terminologique

Nous allons étudier les différences aux niveaux syntaxique et sémantique.

a) Comparaison syntaxique

Le langage hybride défini par [Nebel90] se décompose en terminologique et assertionnel. Le Tableau 1 donne le parallèle entre les termes utilisés dans ces deux langages.

Langage hybride	Langage avec références aux instances
concept	classe
rôle	attribut de classe
objet	instance
relation	attribut d'instance

Tableau 1 : parallèle des termes utilisés dans le langage hybride et dans le langage avec références aux instances.

Nebel distingue les concepts définis des concepts primitifs ; ces derniers correspondent à l'introduction d'une classe par une relation de spécialisation. Les concepts définis peuvent être définis :

- par conjonction de concepts : ce qui est possible dans notre langage en définissant plusieurs super-classes pour une classe donnée,
- par restriction de valeur d'un rôle donné : ce qui correspond à la restriction de l'attribut d'une classe ou
- par restriction du nombre d'objets qui pourront y appartenir :ce qui n'a pas d'équivalent dans notre langage.

Il n'autorise pas non plus les cycles dans les définitions des concepts car il n'est pas possible selon lui d'en définir le sens. Une différence superficielle est que concept, objet et rôle sont définis dans le langage terminologique en une seule fois, alors que la définition de classe et objet peut être composée de plusieurs assertions dans notre langage ; cela nécessite alors de faire appel à des opérateurs pour modifier les concepts et les objets, même si les modifications laissent la base consistante.

Une autre différence porte sur la distinction entre rôles et attributs : un rôle est défini comme une relation, alors qu'un attribut est une fonction. De plus, les rôles sont ouverts, ce qui n'est pas le cas des attributs.

b) Comparaison sémantique

La sémantique que nous avons définie est inspirée de celle de Nebel, donc les idées principales sont forcément identiques :

- un concept est défini comme un ensemble d'objets
- un objet appartient à un concept si l'interprétation de l'objet appartient à celle du concept

Cependant, on peut noter une différence cruciale qui est que l'interprétation d'un concept est *égal à* l'ensemble des objets qui satisfont les contraintes posées sur le concept, alors que l'interprétation d'une classe est *inclue* dans cet ensemble d'objets. Ceci nous permet de distinguer une instance qui *peut* appartenir à une classe, d'une instance qui *est* rattachée à une classe et ainsi de laisser ce choix de rattachement à l'utilisateur. La seule digression en est la règle de ($\in$-inférence) qui force le rattachement d'une instance à une classe, dans un souci de préservation de la consistance de la base autant que faire se peut.

2. TROPES

Nous avons défini le langage dans l'optique de définir la révision qui est implémentée pour TROPES, donc ils sont très proches. Notons cependant les différences :

- la multi-spécialisation est autorisée dans le langage proposé mais pas dans TROPES ;
- la multi-instanciation n'est pas aussi large dans TROPES : la notion de point de vue intervient et une instance ne peut être rattaché qu'à une seule classe par point de vue ;

- la définition des attributs est plus élaborée en TROPES : les attributs peuvent être multi-valués, c'est-à-dire admettre un ensemble de valeurs comme valeur d'attribut ;
- TROPES autorise les contraintes, ainsi que les passerelles entre points de vue, ce qui se traduirait par des inclusions d'interprétations de classes.

Sur le sous-langage commun à TROPES et au langage défini ici, la sémantique est identique.

D. Conclusion

Nous avons défini une formalisation qui s'applique aux langages d'objets qui utilisent :

- des objets,
- des classes comme étant des ensembles d'objets définis en intension,
- de relation de spécialisation entre classes
- des attributs d'objets valués par une valeur de type ou par une autre instance.

En particulier, la sémantique de tels langages a été clairement explicitée, ce qui permet de lever toute ambiguïté sur l'interprétation du contenu des bases.

De plus, un système déductif – correct et complet pour les bases consistantes – permet de manipuler les bases sans utiliser les modèles. La déduction peut ainsi être implémentée par un algorithme de complexité polynomiale.

Enfin, la forme première d'une base est l'ensemble minimal fini d'assertions qui est suffisant pour déduire toute la base. La minimalité de la forme première garantit que les assertions supprimées ne pourront plus être déduites. Cette notion sera utile lors de la révision, car celle-ci nécessite la suppression d'assertions de la base, cette suppression devant être effective, pour que la base obtenue après l'ajout de la dernière connaissance soit consistante.

Il est maintenant possible de décrire un mécanisme de révision spécifiquement adapté au langage d'objets décrit ici.

Chapitre III. Révision dans un langage d'objets

La syntaxe et la sémantique d'un langage d'objets ont été définies au chapitre précédent. Il s'agit à présent d'étudier les possibilités de réviser une base de connaissance quand un ajout est inacceptable dans cette base. La première étape est la caractérisation de ce qu'est un ajout inacceptable. Nous verrons qu'en plus des inconsistances définies classiquement, un autre problème spécifique aux langages d'objets se pose : l'existence de classes, appelées incohérentes, ne pouvant contenir d'instance (section A). La définition proposée pour les bases révisées correspond à celle donnée par [Alchourròn&85], mais elles sont exprimées sous une forme particulière utilisant la forme première (section B). L'une des particularités de cette étude est de déterminer un *ensemble* de bases révisées et non une seule ; le critère de choix entre les différentes bases révisées possibles n'étant pas formalisé, ce choix est laissé au concepteur de la base de connaissance.

Cependant, il s'agit de restreindre au maximum les possibilités en ne proposant que les bases qui modifient le moins la base initiale (section C). Cette modification minimale est définie de manière sémantique et syntaxique grâce à une relation d'ordre entre bases révisées. De plus, nous donnons des caractérisations sémantique et syntaxique des bases contractées minimales. Enfin, afin de pouvoir déterminer de manière pratique les bases contractées qui sont minimales, la minimalité des bases contractées est caractérisée en fonction de ses assertions (paragraphe C.2).

La révision proposée est alors examinée selon les critères donnés dans le paragraphe I.F.1.b (section D) et ses principales caractéristiques sont résumées (section E). La mise en œuvre algorithmique est donnée dans le chapitre IV et l'étude de la complexité en annexes C et D.

A. Inconsistance et incohérence

Nous étudions les deux notions d'inconsistance et d'incohérence, en en donnant une caractérisation syntaxique pour le langage simple puis pour le langage admettant les références aux instances.

1. D'un point de vue sémantique

En utilisant la notion de modèle, la définition de l'inconsistance est immédiate et identique à celle des formalismes logiques. Rappelons cette définition donnée dans le paragraphe II.A.2.b :

DÉFINITION 8 : BASE INCONSISTANTE

Une base *B* est inconsistante ssi il n'existe pas de modèle de *B*.

Cependant, cette notion ne peut être utilisée concrètement, car la détermination des modèles d'une base ou de la non existence de modèle est trop complexe, en particulier à cause de leur nombre et de leur taille. C'est pourquoi nous nous intéressons à une caractérisation syntaxique de l'inconsistance dans le paragraphe 2, afin de pouvoir définir un algorithme.

D'autre part, l'inconsistance n'englobe pas tous les problèmes qui peuvent apparaître dans les bases : nous utilisons la notion de classe incohérente. Cette notion permet de détecter les classes qui ne peuvent avoir d'instance dans aucun des modèles, comme celle de la Figure 23. Cette notion a été introduite dans [Nebel90, paragraphe 3.2.3]. Ces classes représentent des potentiels d'inconsistance car les assertions de rattachement d'une instance à ces classes rendent la base inconsistante, et vu qu'elles ne peuvent contenir d'instances, leur utilité semble bien faible.

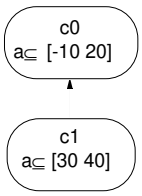


Figure 23 : Exemple de base consistante contenant une classe incohérente (c_1).

DÉFINITION 20 : CLASSE INCOHÉRENTE

Une classe *c* est incohérente dans une base consistante *B* ssi son interprétation est vide dans tout modèle de *B*.

DÉFINITION 21 : BASE INCOHÉRENTE

Une base consistante *B* est incohérente ssi elle contient une classe incohérente ; une base *B* est cohérente ssi elle ne contient aucune classe incohérente.

Les bases acceptables sont alors des bases qui ne contiennent ni inconsistance, ni classe incohérente.

DÉFINITION 22 : BASE ACCEPTABLE, BASE INACCEPTABLE

Une base B est acceptable ssi elle est consistante et cohérente. Dans le cas contraire, elle est inacceptable.

DÉFINITION 23 : ASSERTION ACCEPTABLE/INACCEPTABLE DANS UNE BASE

Une assertion δ est acceptable (resp. inacceptable) dans une base B ssi la base $B \cup \{\delta\}$ est acceptable (resp. inacceptable).

2. D'un point de vue syntaxique

La définition sémantique de base inconsistante et de classe incohérente est insuffisante pour une détection efficace de ces deux types de problème. En effet, d'une part il existe généralement une infinité de modèles de taille infinie. D'autre part, l'objectif étant de réviser la base, il est nécessaire de pouvoir localiser inconsistance et incohérence, c'est-à-dire de déterminer les ensembles d'assertions qui les provoquent. Ceci est envisageable car toutes les assertions ne sont pas déductibles d'une base inconsistante, contrairement aux langages logiques classiques.

Nous donnons donc ci-dessous une caractérisation syntaxique de base inconsistante et de classe incohérente. Nous distinguons une fois de plus le langage simple du langage avec références aux instances, toujours pour faciliter la compréhension, le second cas étant plus général que le premier.

a) Dans le langage simple

(1) Inconsistance

L'inconsistance se traduit de manière syntaxique par la présence d'ensembles particuliers d'assertions. Cette expression syntaxique est donc plus délicate que la définition sémantique, mais elle permet de cibler les assertions responsables. En fait, une base peut être inconsistante pour trois motifs distincts qui se traduisent par trois ensembles différents d'assertions.

DÉFINITION 24 : MOTIF D'INCONSISTANCE (LANGAGE SIMPLE)

Un motif d'inconsistance dans une base B est l'un des trois ensembles d'assertions suivants :

(1) $\{o.a=v,\ o.a=v'\}$ tel que $(o.a=v) \in B$, $(o.a=v') \in B$ avec $v \neq v'$, $o \in O$, $a \in A$, $v, v' \in V$.

(2) $\{o \in c,\ c.a \subseteq d,\ o.a=v\}$ tel que $B \vdash o \in c$, $B \vdash c.a \subseteq d$, $(o.a=v) \in B$ avec $\neg v : d$, $o \in O$, $c \in C$, $a \in A$, $d \in T$ et $v \in V$.

(3) $\{o \in c_i\}_i \cup \{c_i.a \subseteq d_i\}_i$ tel que pour tout i, $B \vdash o \in c_i$, $B \vdash c_i.a \subseteq d_i$ avec $\bigwedge_i d_i = \bot$, $o \in O$, $c_i \in C$, $a \in A$.

PROPOSITION 18 : INCONSISTANCE SYNTAXIQUE

B est inconsistante ssi elle contient au moins un motif d'inconsistance.

Le motif (1) indique que deux valeurs différentes ne peuvent être définies pour le même attribut d'instance. Le motif(2) indique que lorsqu'une valeur est définie pour un attribut d'instance, il y a inconsistance si la valeur ne satisfait pas les contraintes posées sur l'une de ses classes d'appartenance (Figure 24). Le motif (3) reflète le cas où aucune valeur n'est possible pour un attribut d'instance car les restrictions posées par ses classes d'appartenance ne sont pas compatibles (Figure 25).

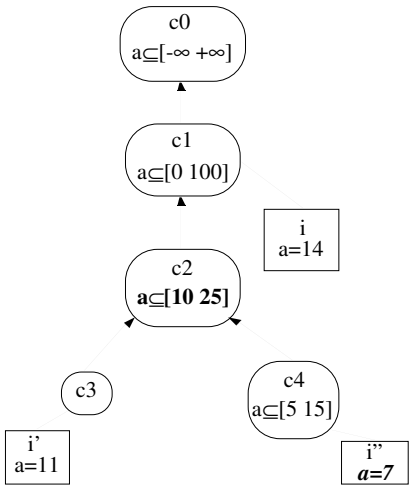


Figure 24 : Exemple de base inconsistante (motif 2) - repris du chapitre II.A.2. $(i''.a=7) \in B$, $(i'' \in c_4) \in B$ $(c_2.a \subseteq [10\ 25]) \in B$ et $(c_4 \leq c_2) \in B$ donc, d'après la règle de (d-héritage), $B \vdash c_4.a \subseteq [10\ 25]$ et $\neg 7 : [10\ 25]$.

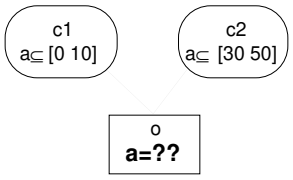


Figure 25 : Exemple de base inconsistante (motif 3). $(o \in c_1) \in B$, $(o \in c_2) \in B$, $(c_1.a \subseteq [0\ 10]) \in B$, $(c_2.a \subseteq [30\ 50]) \in B$ et $[0\ 10] \wedge [30\ 50] = \bot$.

(2) Incohérence

De même qu'il existe une définition syntaxique de l'inconsistance, il est intéressant de caractériser syntaxiquement l'incohérence pour la détecter et l'éviter.

PROPOSITION 19 : INCOHÉRENCE SYNTAXIQUE

$c \in C$ est une classe incohérente dans la base B ssi il existe $a \in A$ tels que $B \vdash c.a \subseteq \bot$.

b) Dans le langage avec référence aux instances

(1) Inconsistance

Cette notion recouvre des motifs plus divers que dans le cas du langage simple. Par exemple, la base de la Figure 17 de la section II.B à laquelle est simplement ajoutée l'assertion ($o_1.b=o_4$) est inconsistante (Figure 26). En effet, $\{c_4 \leq c_3$; $o_1 \in c_4$; $c_3.b \subseteq c'_1$; $o_4 \in c'_2$; $c'_1.a' \subseteq [1\ 20]$; $c'_2.a' \subseteq [30\ 100]\} \subseteq B$. Donc :

$I(o_4)=I(b)(I(o_1)) \in I(c'_1)$, d'où, tous les modèles doivent satisfaire :

$I(a')(I(o_4)) \in I(a'|c'_1)) \cap I(a'|c'_2))=\emptyset$, ce qui est impossible.

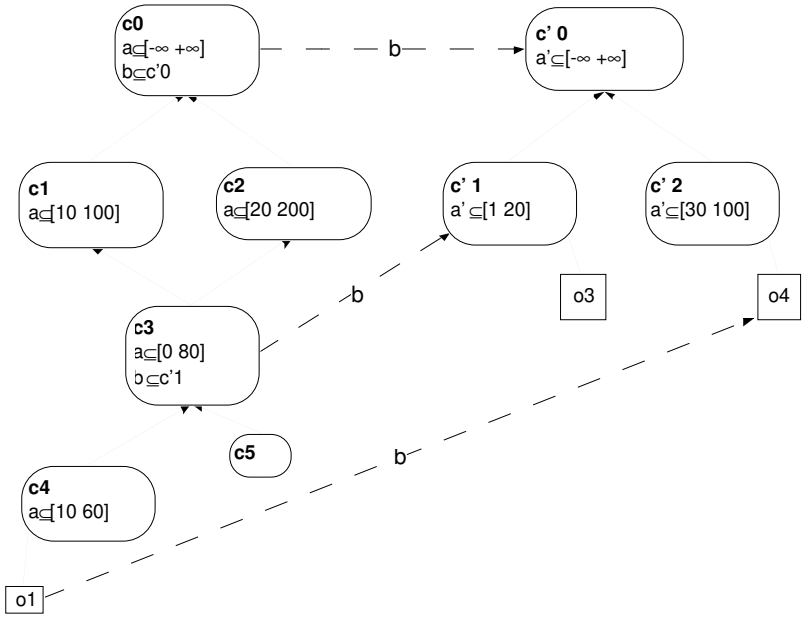


Figure 26 : Exemple de base inconsistante. L'instance o_1 référence l'instance o_4 qui doit donc appartenir à la classe c'_1 à cause de la restriction posée sur la classe c_3 ; or les restrictions posées sur l'attribut a' dans les classes c'_1 et c'_2 sont incompatibles donc il y a inconsistance.

Il faut remarquer ici que ce qui rend la base inconsistante est l'incompatibilité des contraintes posées sur les classes c'_1 et c'_2, et non le fait que l'instance o_4 ne soit pas rattachée explicitement à une sous-classe de la classe c'_1. En effet, une inconsistance n'apparaît que s'il

existe des connaissances contradictoires dans la base, ce qui est le cas lorsqu'une instance *ne peut pas* être attachée à une classe et non si elle *n'est pas* attachée à la classe en question. Ainsi, si la restriction de domaine de la classe c'_2 avait été $c'_2.a' \subseteq [10\ 100]$, la base aurait été consistante, car c'_1 et c'_2 n'auraient pas posées de restrictions incompatibles sur l'attribut a.

L'expression des motifs d'inconsistance est similaire à celle dans le langage simple.

DÉFINITION 25 : MOTIF D'INCONSISTANCE (LANGAGE COMPLEXE)

Un motif d'inconsistance dans une base B est l'un des trois ensembles d'assertions suivants :

(1) $\{o.a=v, o.a=v'\}$ tel que $(o.a=v) \in B$, $(o.a=v') \in B$ avec $v \neq v$, $o \in O$, $a \in A$, $v, v' \in V \cup \boldsymbol{O}$.

(2) $\{o \in c,\ c.a \subseteq d,\ o.a=v\}$ tel que $B \vdash o \in c$, $B \vdash c.a \subseteq d$, $(o.a=v) \in B$ avec $\neg v : d$, $o \in O$, $c \in C$, $a \in A$, $d \in T$ et $v \in V$.

(3) $\{o \in c_i\}_i \cup \{c_i.a_1.a_2.\ldots.a_n \subseteq *d_{ij}\}_i$ tel que pour tout i, $B \vdash o \in c_i$, $(c_i.a_1.a_2.\ldots.a_n \subseteq *d_{ij}) \in B$ avec $\wedge_{ij} d_{ij} = \perp$, $o \in O$, $c_i \in C$, $a_i \in A$.

Le motif (1) est étendu aux attributs d'instance dont la valeur est une autre instance. Nous considérons que deux instances sont différentes si leur nom est différent, car l'hypothèse de nom unique est posée dans le domaine.

Par rapport au langage simple, le motif (2) est identique mais a des applications différentes. En effet, l'assertion $(o \in c)$ peut être déduite grâce à la règle de ($\in$-inférence), ce qui permet une plus large diffusion de l'inconsistance par rapport à l'assertion ajoutée dans la base (Figure 27).

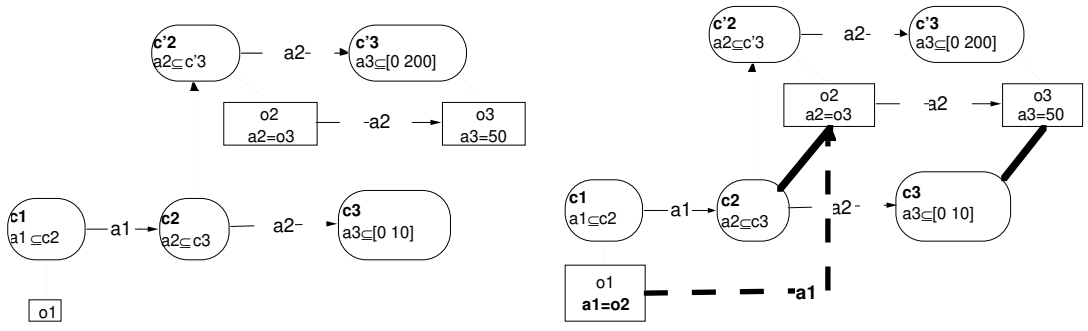


Figure 27 : Si on ajoute $o_1.a_1=o_2$ dans la base (a), la règle de ($\in$-inférence) permet de déduire $(o_2 \in c_2)$ et donc $(o_3 \in c_3)$ (b), d'où inconsistance d'après le motif (2) entre la valeur de $o_3.a_3$ (50) et les restrictions posées sur cet attribut dans sa nouvelle classe d'appartenance ([0 10]). En utilisant la formulation (2'), on a $c_1.a_1.a_2.a_3 \subseteq *[0\ 10]) \in B$, $(o_1.a_1.a_2.a_3 = *50) \in B$, $(o_1 \in c_1) \in B$ et $\neg 50 : [0\ 10]$.

Il est également intéressant de pouvoir définir un motif d'inconsistance en fonction non pas d'assertions déduites, mais d'assertions appartenant à la base ; dans cette optique, le second motif peut alors s'écrire :

(2') $\{o \in c,\ o.a_1.a_2.\ldots.a_n = *v,\ c.a_1.a_2.\ldots.a_n \subseteq *d\}$ tel que $(c.a_1.a_2.\ldots.a_n \subseteq *d) \in B$, $(o.a_1.a_2.\ldots.a_n = *v) \in B$, $(o \in c) \in B$ avec $\neg v : d$, $o \in O$, $a_i \in A$, $v \in V$, $d \in T$ et $n \geq 1$.

Le troisième motif doit tenir compte des chemins d'attributs. En effet, pour déduire qu'une instance ne peut appartenir à un ensemble de classes, il faut qu'il existe une incompatibilité sur des attributs qui sont restreints par un domaine et non par une classe, et donc parcourir des chemins d'attributs. Cette formulation permet de faire intervenir des restrictions de classes qui appartiennent à la base et qui ne sont pas déduites. Illustrons ce motif par quelques exemples :

- Dans le cas de la Figure 26, $B \vdash o_4 \in c'_1$, par les règles de (d-héritage) et de ($\in$-inférence) et $B \vdash o_4 \in c'_2$, car $(o_4 \in c'_2) \in B$. Le motif d'inconsistance est complété par $(c'_1 \subseteq *[1\ 20]) \in B$ et $(c'_2 \subseteq *[30\ 100]) \in B$.
- Dans la Figure 28, la base est inconsistante car l'instance o est attachée à une classe incohérente : $(c_3.b.a' \subseteq *[1\ 20]) \in B$ et $(c_3.b.a' \subseteq *[30\ 40]) \in B$

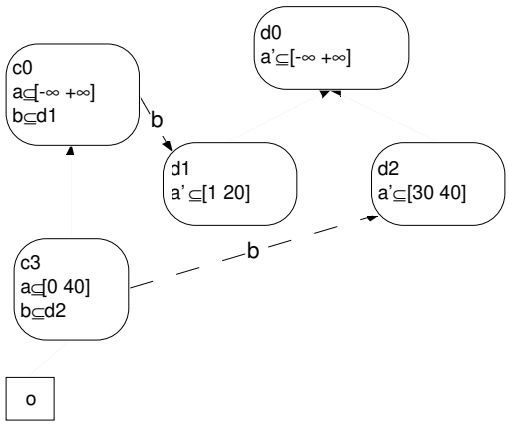


Figure 28 : *o* est attaché à une classe incohérente (c'est un cas particulier du cas suivant).

- L'instance est attachée à un ensemble de classes qui posent des restrictions incompatibles (Figure 29).

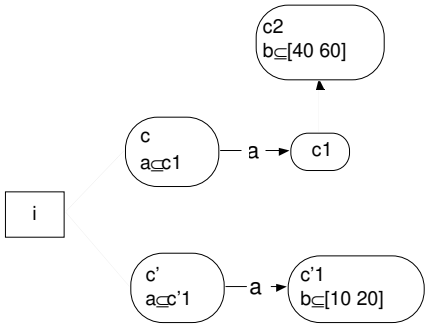


Figure 29 : cette base est inconsistante car *i* est attaché à *c* et *c'* et que $(c.a.b \subseteq *[40\ 60]) \in B$ et $(c'.a.b \subseteq *[10\ 20]) \in B$.

(2) Incohérence

Lorsqu'un attribut de classe est restreint par des classes dont l'intersection des restrictions sur l'un des attributs est vide, la classe est incohérente. Comme les références entre classes ne sont pas cycliques, cette notion d'incohérence est bien fondée.

L'incohérence ne peut se détecter de manière syntaxique qu'en obtenant un domaine vide ; cette détection n'est possible que sur un chemin d'attribut qui se termine par des domaines. En effet, pour affirmer qu'une classe est vide, ou plus justement que son interprétation est vide, il faut qu'aucune valeur ne soit possible pour l'un de ses attributs. Pour les attributs qui référencent des classes, il n'est pas possible de dire s'il existe au moins une valeur possible sans étudier les attributs de ces classes. C'est pourquoi, la détermination de l'incohérence se fait en suivant un chemin d'attributs aboutissant à un type simple (le dernier attribut de la chaîne est restreint par un domaine).

La Proposition 20 permet de refléter différents cas qui sont illustrés par les figures 30 et 31.

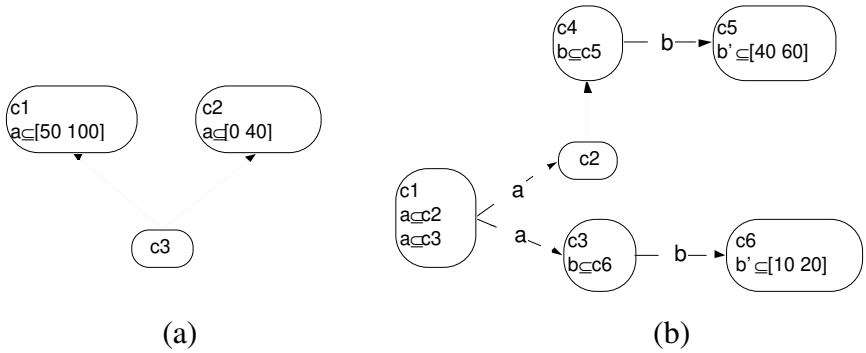


Figure 30 : (a) La classe c_3 est incohérente car elle hérite de deux restrictions d'attribut dont l'intersection est vide. (b) La classe c_1 est incohérente car elle référence deux classes qui ne sont pas compatibles ($c_1.a.b.b' \subseteq *[10 \supseteq 20] \in B$ et $c_1.a.b.b' \subseteq *[40\ 60] \in B$)

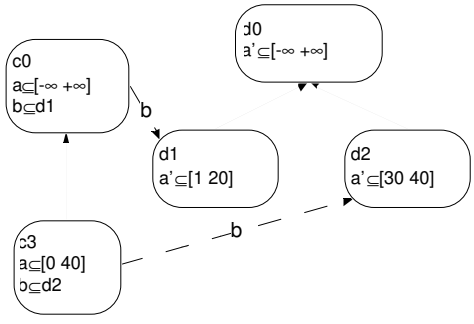


Figure 31 : la classe c_3 est incohérente car l'intersection des classes référencées est vide, en effet c_3 hérite de la restriction de la classe c_0, et donc $c_3.b.a' \subseteq *[1\ 20] \in B$ et $c_3.b.a' \subseteq *[30\ 40] \in B$.

PROPOSITION 20 : INCOHÉRENCE SYNTAXIQUE

Soit B une base consistante, $c \in C$ est une classe incohérente si et seulement si il existe a_1, a_2, …, $a_n \in A$, $n \geq 1$, tels que pour tout $1 \leq l \leq k$, $(c.a_1.a_2.\ldots.a_n \subseteq^* d_l) \in B$, $d_l \in T$ et $\bigwedge_{1 \leq l \leq k} d_l = \perp$.

3. Localité de la déduction en cas d'inconsistance ou d'incohérence

Dans un formalisme logique classique, dès qu'une base contient une formule et son contraire, n'importe quelle formule est déductible de cette base. Ceci n'est pas le cas dans le langage décrit dans le chapitre II car les règles de déduction ne permettent pas de déduire n'importe quelle assertion à partir d'une base inconsistante. Par contre, au niveau sémantique, toutes les assertions sont des conséquences d'une base inconsistante. C'est une raison supplémentaire pour se placer au niveau syntaxique pour étudier les bases inconsistantes.

La localité de la déduction dans le cas de base inconsistante se traduit par le fait que la présence d'un motif d'inconsistance ne permet pas de déduire d'assertions farfelues, c'est-à-dire d'assertions qui ne sont plus déductibles si *l'une quelconque* des assertions du motif n'est pas présente dans la base. L'ajout d'une assertion dans la base permet évidemment de déduire de nouvelles assertions, mais si elle déclenche une inconsistance, les déductions sont les mêmes que si l'ajout avait été consistant.

Cette propriété est vérifiée à une exception près :supposons qu'il existe o, $\{c_i\}$, a tels que $B \vdash o \in c_i$, $B \vdash c_i.a \subseteq d_i$ et $\bigwedge_i d_i = \perp$; il y a inconsistance. Dans ce cas, par la règle de (d-clôture), $B \vdash c_i.a \subseteq \{v\}$ pour tout $v \in V$ et donc par la règle de (o-valuation), $B \vdash o.a = v$ pour tout $v \in V$, ce qui n'est pas très justifié (Figure 32).

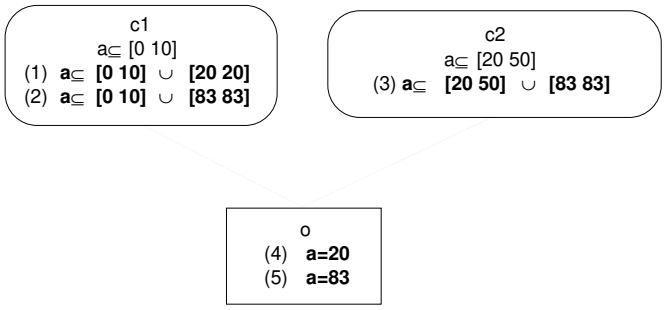


Figure 32 : Exemple de base inconsistante dans laquelle des assertions sont déduites grâce à un motif d'inconsistance ; les assertions (1), (2) et (3) sont déduites par la règle de (d-clôture) ; (4) est déduite par (o-valuation) grâce à la restriction de l'attribut a posée sur *c1* (1) et à la restriction de l'attribut a de la classe *c2*(3) ; (5) est déduite par (o-valuation) grâce à (2) et à (3).

L'effet de ces déductions non voulues est cependant limitée car ces nouvelles assertions ne font pas apparaître de nouveaux motifs d'inconsistance : en particulier, les assertions de la forme ($o.a=v$) qui apparaissent dans les motifs de type (2) doivent appartenir à la base.

De plus, l'ajout d'une assertion occasionne de nouvelles déductions relativement localisées ; en général, il s'agit d'assertions concernant la même instance ou la même classe, éventuellement des classes de la même hiérarchie. On peut noter tout de même le cas de la règle de ($\in$-inférence) qui permet de déduire des assertions concernant une autre instance, ce qui peut conduire à une propagation à un ensemble d'instances. Donc, lorsqu'une assertion ajoutée produit une inconsistance, les motifs d'inconsistance contiennent l'assertion ajoutée ou tout du moins une assertion déductible directement d'elle. De manière imagée, l'inconsistance ne se propage pas et ne touche pas l'ensemble de la base.

Dans le cas d'une classe incohérente, l'incohérence est propagée à toutes ses sous-classes et à toutes les classes dont l'un des attributs les référencent. Des instances attachées à l'une de ces classes incohérentes déclenchent une inconsistance par des motifs de type (3) ; si aucune instance n'est attachée à ces classes, la base est consistante. Les incohérences ont été déclenchées par l'ajout d'une assertion et il est donc aisé de repérer l'ensemble de classes incohérentes qui provoquent toutes les autres incohérences. Lorsqu'une classe c est incohérente à cause d'un attribut a, toutes les assertions ($c.a \subseteq d$) avec d quelconque sont déductibles. Cependant, ceci est vrai aussi bien dans le cas d'une base acceptable que d'une base inacceptable et ces nouvelles assertions déduites ne rendent pas d'autres classes incohérentes.

4. Comparaison avec la logique

La définition sémantique de l'inconsistance est la même que dans le cadre logique : une base est inconsistante s'il n'existe pas de modèle de cette base. Cependant, cette notion de consistance est complétée par celle de cohérence, qui permet de détecter les bases possédant des classes ne pouvant pas contenir d'instances, classes qui ne sont pas utiles pour la modélisation du domaine. En effet, elles ne peuvent résulter que d'une erreur de la part du concepteur, vu qu'aucune interprétation ne peut être donnée à de telles classes.

Cependant, dans les formalismes logiques, une base inconsistante permet de tout déduire. Contrairement, dans le langage d'objets décrit, la présence d'une inconsistance ou d'une incohérence ne se propage pas à toute la base, c'est-à-dire que la base ne permet pas de déduire n'importe quelle assertion, même si des assertions, du type ($o.a=v$) en particulier, peuvent être déduites sans réelles justifications (cf. 3).

De plus, dans ce travail, la caractérisation syntaxique de base inconsistante et de classe incohérente permet de ne pas utiliser la notion de modèle pour les détecter. Cela permet d'avoir un algorithme de coût polynomial (voir l'étude de la complexité en annexes C et D) et surtout de localiser précisément la partie de la base concernée par l'inconsistance ou l'incohérence, afin de faciliter la révision. Cette localisation des problèmes se rapproche des techniques d'abduction en logique (voir par exemple [Kakas&90, Aravindan&94]), mais, encore une fois, la structure de la base permet de mieux guider la détermination des assertions fautives et l'absence de connecteur de disjonction entre les assertions évite l'explosion combinatoire du nombre d'assertions intervenant dans l'inconsistance.

B. Base révisée

La section précédente a donné une caractérisation syntaxique de l'inconsistance et de l'incohérence. Lorsque l'ajout d'une assertion dans la base produit une inconsistance ou une incohérence, il s'agit de réviser la base pour que l'ajout puisse se faire tout en préservant la consistance de la base. Cette section se consacre à la *définition* d'une base révisée par les contraintes qu'elle doit satisfaire ; l'*obtention* des bases révisées est détaillée dans les sections décrivant l'algorithme (chapitre IV sections A, B et C).

Comme les propositions sur la complétude et les propriétés de la forme première sont identiques que les bases soient exprimées dans le langage d'objets simple ou dans le langage avec référence aux instances, il ne sera pas fait de distinction entre les deux langages dans cette section. La différence entre les caractérisations de l'inconsistance et de l'incohérence n'est pas sensible non plus lors de la définition de la base révisée, mais elle l'est pour la description de l'algorithme.

La révision s'effectue sur des bases acceptables, c'est-à-dire qui ne contiennent ni contradiction, ni classe incohérente (Définition 22). La révision est définie pour l'ajout de toutes les assertions différentes de ($c.a \subseteq \bot$), car aucune base cohérente ne peut contenir cette assertion.

DÉFINITION 26 : ASSERTION COHÉRENTE
Une assertion est cohérente si elle ne peut pas s'écrire sous la forme ($c.a \subseteq \bot$).

Une base révisée est définie en deux étapes :

- une contraction de la base initiale afin d'obtenir une sous-base consistante avec l'assertion à ajouter ;
- l'ajout de l'assertion dans la sous-base.

La contraction d'une base n'est pas unique, c'est pourquoi nous définissons, non pas un opérateur de contraction, comme cela est fait en général dans la littérature [par exemple Alchourròn85, Dalal88, Nebel92], mais des bases contractées.

DÉFINITION 27 : BASE CONTRACTÉE

Soient B une base acceptable et δ une assertion cohérente telle que $B \cup \{\delta\}$ soit inacceptable. B' est une base contractée de (B,δ) ssi $B' \cup \{\delta\}$ est acceptable et $Cn(B') \subset Cn(B)$, ce qui se traduit sémantiquement par $B \models B'$.

En cas d'inconsistance ou d'incohérence, cette définition permet de se replier sur une base B' plus réduite que B, en considérant les clôtures déductives, et compatible avec l'ajout de δ. Cette définition est évidemment très faible, car la base vide est une base contractée de n'importe quel couple (B,δ). Rappelons que nous ne considérons pas de critère de minimalité pour l'instant ; ceci sera traité dans le paragraphe C.

Il existe un ensemble de bases contractées qui ont la même clôture déductive. Dans la suite, on ne considérera plus que l'ensemble des bases contractées quotienté par la relation d'équivalence «avoir la même image par Cn (ou par FP)». L'expression «base contractée» désignera donc la représentante sous forme première d'une classe de bases contractées.

DÉFINITION 28 : BASE RÉVISÉE

Soient B une base acceptable et δ une assertion cohérente telle que $B \cup \{\delta\}$ soit inacceptable. B' est une base révisée de (B,δ) ssi $B' = B'' \cup \{\delta\}$ avec B'' une base contractée de (B,δ).

Afin de pouvoir manipuler les bases contractées facilement, elles sont exprimées sous une forme donnant les assertions supprimées de la forme première (B'_-) — qui ne seront effectivement plus déductibles grâce à la définition de la forme première (Définition 12 dans le chapitre II) — et certaines assertions (B'_+), déductibles de B mais pas de $FP(B) - B'_-$, qui sont conservées. Pour avoir une forme plus simple, l'intersection de ces deux ensembles est vide, sinon des assertions seraient retranchées et ajoutées de nouveau.

Toute base contractée de (B,δ) B' peut se caractériser de manière unique par :
$B' = FP(B') = (FP(B) - B'_-) \cup B'_+$ avec comme contraintes :

- $B'_- \cap B'_+ = \varnothing$,
- $B'_- \subseteq FP(B)$ et
- $B'_+ \subseteq Cn(B)$.[16]

[16] Cette écriture n'est pas spécifique aux bases contractées; toute sous-base de Cn(B) peut s'exprimer ainsi.

La Figure 33 explicite ce que contient une base contractée en fonction de B'_+ et B'_-.

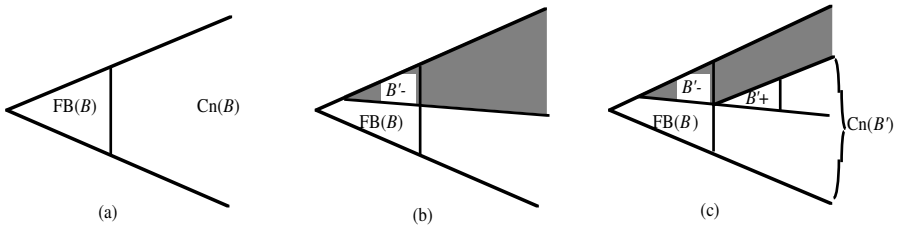


Figure 33 : La forme première FP(B) permet de déduire la clôture déductive Cn(B) (a) ; le retrait des assertions appartenant à B'_- permet d'obtenir une base acceptable, mais ne rend plus possible la déduction d'un certain nombre d'assertions de Cn(B) (b) ; parmi ces assertions, certaines sont consistantes avec l'ajout envisagé et peuvent donc être ajoutées ; c'est l'ensemble B'_+ (c).

Dans B'_+ il n'y a que les assertions strictement nécessaires. Les assertions redondantes avec d'autres assertions de Cn(FP(B)$-B'_-$) ne sont pas autorisées et il n'y a pas d'assertions redondantes dans B'_+, puisque la base contractée est exprimée sous forme première.

La contraction d'une base de connaissance consiste donc à supprimer des assertions de la forme première et à en ré-ajouter certaines qui étaient déductibles. L'algorithme, donné au chapitre IV, décrit précisément comment les bases révisées sont obtenues, mais, en simplifiant, une base contractée est obtenue en combinant les stratégies suivantes :

- supprimer des assertions du type ($o.a$=v) ou ($o.a$=o') ;
- remplacer des assertions du type ($i \in c$) par ($i \in c'$) avec ($c \leq^* c'$)$\in$ FP(B) ;
- remplacer des assertions du type ($c.a \subseteq d$) par ($c.a \subseteq d'$) avec $d <: d'$;
- remplacer des assertions du type ($c.a \subseteq c'$) par ($c.a \subseteq c''$) avec ($c' \leq^* c''$)$\in$ FP(B) ;
- remplacer des assertions du type ($c \leq c'$) par ($c \leq c''$) avec ($c' \leq^* c''$)$\in$ FP(B).

Remarquons qu'une instance ne peut être déplacée que sous une super-classe de son ancienne classe d'appartenance car la clôture déductive de la base contractée est incluse dans celle de la base initiale.

Voyons des exemples de bases révisées – obtenues par ajout de l'assertion problématique dans les bases contractées – dans le cas du langage simple et du langage complexifié.

EXEMPLE : LANGAGE SIMPLE

La Figure 34 représente graphiquement différentes bases révisées possibles d'un ajout inconsistant dans la base inconsistante de la Figure 24 :

FP(B)={$c_1 \leq c_0$; $c_2 \leq c_1$; $c_3 \leq c_2$; $c_4 \leq c_2$; $i \in c_2$; $i' \in c_3$; $i'' \in c_4$; $c_0.a \subseteq [-\infty \ +\infty]$; $c_1.a \subseteq [0\ 100]$; $c_2.a \subseteq [10\ 25]$; $c_4.a \subseteq [10\ 15]$; $i.a$=14 ; $i'.a$=11}

avec δ=($i''.a$=7).

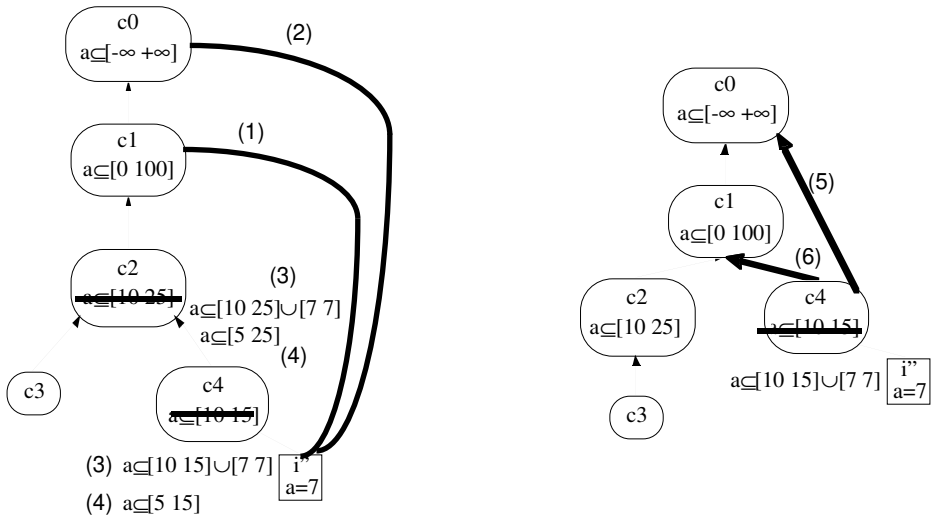


Figure 34 : exemples de bases révisées possibles : (1) et (2) : l'instance est déplacée sous une autre classe en supprimant son appartenance à sa classe d'origine et en l'attachant à l'une de ses super-classes. (3) et (4) : deux assertions de restriction de domaine sont supprimées ; elle peuvent être remplacées par une autre, avec un domaine plus grand que le précédent. (5) et (6) : la classe est déplacée en supprimant une assertion de spécialisation et en la remplaçant par d'autres et sa restriction de domaine est modifiée.

Selon la notation donnée, les différentes bases révisées illustrées sur la Figure 34, exprimées sous forme première, se détaillent comme suit :

$B_1 = \mathrm{FP}(B) - \{i'' \in c_4\} \cup \{i'' \in c_1\} \cup \{i''.a=7\}$

$B_2 = \mathrm{FP}(B) - \{i'' \in c_4\} \cup \{i'' \in c_0\} \cup \{i''.a=7\}$

$B_3 = \mathrm{FP}(B) - \{c_2.a \subseteq [10\ 25]\ ;\ c_4.a \subseteq [10\ 15]\}$

$\quad \cup \{c_2.a \subseteq [10\ 25] \cup [7\ 7]\ ;\ c_4.a \subseteq [10\ 15] \cup [7\ 7]\} \cup \{i''.a=7\}$

$B_4 = \mathrm{FP}(B) - \{c_2.a \subseteq [10\ 25]\ ;\ c_4.a \subseteq [10\ 15]\} \cup \{c_2.a \subseteq [5\ 25]\ ;\ c_4.a \subseteq [5\ 15]\} \cup \{i''.a=7\}$

$B_5 = \mathrm{FP}(B) - \{c_4 \leq c_2\ ;\ c_4.a \subseteq [10\ 15]\} \cup \{c_4 \leq c_0\ ;\ c_4.a \subseteq [10\ 15] \cup [7\ 7]\} \cup \{i''.a=7\}$

$B_6 = \mathrm{FP}(B) - \{c_4 \leq c_2\ ;\ c_4.a \subseteq [10\ 15]\} \cup \{c_4 \leq c_1\ ;\ c_4.a \subseteq [10\ 15] \cup [7\ 7]\} \cup \{i''.a=7\}$

Il existe beaucoup d'autres bases révisées. Ainsi, les bases obtenues à partir de B_4 en remplaçant $c_2.a \subseteq [5\ 25]$ par $c_2.a \subseteq [2\ 25]$ ou $c_2.a \subseteq [0\ 25]$, sont des bases révisées. De même, des bases telles que $\mathrm{FP}(B) - \{i'' \in c_4\ ;\ c_4 \leq c_2\} \cup \{i''.a=7\}$ sont des bases révisées, mais elles suppriment des assertions de manière superflues. Parmi l'ensemble des bases révisées, certaines ne correspondent pas à des bases qui modifient minimalement la base initiale : cette notion de minimalité est étudiée dans la section suivante.

EXEMPLE : LANGAGE COMPLEXIFIÉ

Étudions les bases révisées suite à l'inconsistance repérée par la Figure 27 (page 90) si l'assertion ($o_1.a_1=o_2$) est ajoutée dans la base.

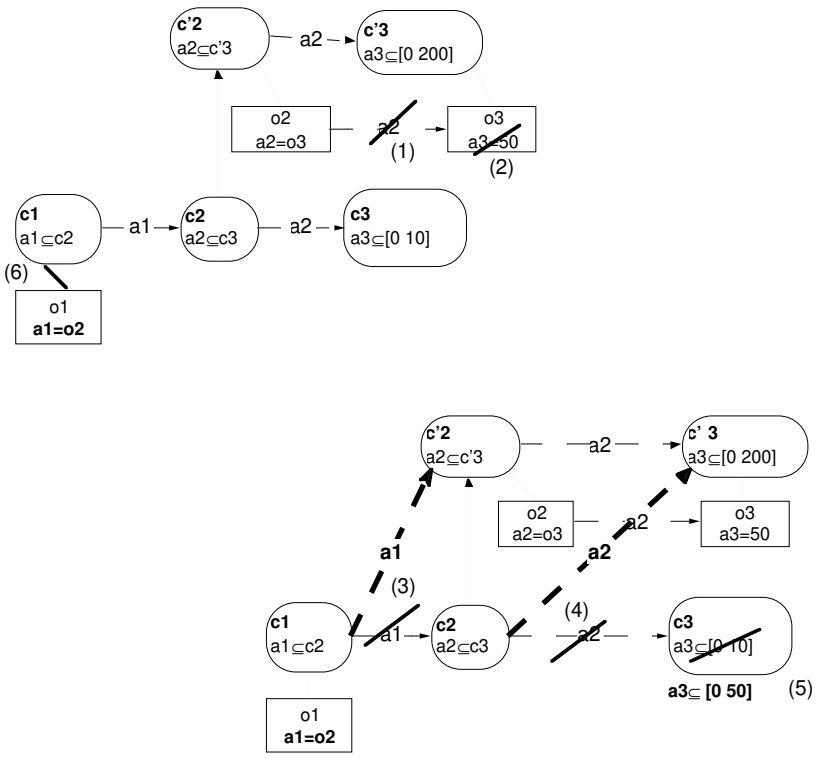


Figure 35 : Suite à l'inconsistance déclenchée par l'ajout de ($o_1.a_1=o_2$), plusieurs bases révisées sont possibles : supprimer ($o_2.a_2=o_3$) (1), ($o_3.a_3$=50) (2), remplacer $c_1.a_1 \subseteq c_2$ par $c_1.a_1 \subseteq c'_2$ (3), remplacer $c_2.a_2 \subseteq c_3$ par $c_2.a_2 \subseteq c'_3$ (4), remplacer $c_3.a_3 \subseteq$[0 10] par $c_3.a_3 \subseteq$[0 50] (5) ou supprimer $o_1 \in c_1$ (6).

Afin que le résultat soit consistant, il est possible de (illustré par la Figure 35) :

- supprimer des valeurs d'attributs des instances : supprimer ($o_2.a_2=o_3$) (1) ou ($o_3.a_3$=50) (2) ;
- modifier des restrictions de domaine : remplacer $c_1.a_1 \subseteq c_2$ par $c_1.a_1 \subseteq c'_2$ (3), remplacer $c_2.a_2 \subseteq c_3$ par $c_2.a_2 \subseteq c'_3$ (4), ou remplacer $c_3.a_3 \subseteq$[0 10] par $c_3.a_3 \subseteq$[0 50] (5) ;
- modifier l'appartenance des instances : supprimer $o_1 \in c_1$ (6).

C. Minimalité

Parmi toutes les bases révisées possibles, certaines ont peut-être perdues trop d'assertions, c'est pourquoi on utilise la notion de minimalité pour ne conserver que celles qui préservent *le plus d'assertions*. Dans la partie I.F, nous avons explicité différents travaux sur la révision et montré la diversité de l'interprétation de la notion de minimalité. La minimalité employée ici est

définie sur les bases contractées de manière équivalente sémantiquement et syntaxiquement. Comme une base révisée est caractérisée de manière unique par la base contractée associée, on parlera de base révisée minimale pour désigner une base révisée associée à une base contractée minimale. Cette relation entre bases révisées ne permet généralement pas de définir une unique plus petite base révisée.

1. Bases contractées minimales

L'ordre entre les bases contractées est défini syntaxiquement et sémantiquement.

DÉFINITION 29 : ORDRE ENTRE LES BASES CONTRACTÉES

Soient B' et B'' deux bases contractées de (B,δ), $B' \propto B''$ ssi les deux conditions équivalentes suivantes sont vérifiées :

(1) $B' \models B''$

(2) $Cn(B'') \subseteq Cn(B')$ (ou tout modèle de B' est modèle de B'')

Ces deux conditions sont effectivement équivalentes grâce au théorème de complétude et au fait que les bases soient consistantes.

Remarquons que la notion de minimalité se fonde sur l'inclusion ensembliste des bases contractées et non sur leur cardinal. En d'autres termes, une base révisée minimale ne préserve pas le plus grand *nombre* d'assertions possibles, mais le plus grand sous-ensemble d'assertions possible de la clôture déductive (rappelons que $Cn(B') \subseteq Cn(B)$).

Les bases contractées minimales sont définies comme les plus petites par rapport à cet ordre.

DÉFINITION 30 : BASE CONTRACTÉE MINIMALE

B' est une base contractée minimale de (B,δ) ssi B' est une base contractée de (B,δ) et qu'il n'existe pas $B'' \neq B'$ telle que B'' soit une base contractée de (B,δ) et $B'' \propto B'$.

Les bases contractées minimales se caractérisent sémantiquement et syntaxiquement.

PROPOSITION 21 : CARACTÉRISATION SÉMANTIQUE DE LA MINIMALITÉ

B' est une base contractée minimale de (B,δ) ssi

$$\forall \gamma \text{ tel que } B \models \gamma \text{ et } B' \cup \{\delta\} \not\models \gamma \text{ alors } B' \cup \{\delta\} \cup \{\gamma\} \text{ est inacceptable.}$$

Intuitivement B' est une base contractée minimale de (B,δ), si aucune autre base contractée de (B,δ) ne permet de satisfaire plus d'assertions de B, au sens de l'inclusion ensembliste.

PROPOSITION 22 : CARACTÉRISATION SYNTAXIQUE DE LA MINIMALITÉ

Si B' et B'' sont deux bases contractées de (B,δ), B' est plus petite que B'' ssi $B'_{\perp} \subseteq B''_{\perp}$ et $B' \vdash B''_{+}$.

Intuitivement, B' est plus petite que B'' si B' a moins d'assertions supprimées et qu'elle permet de déduire tout ce qui a été rajouté dans B''. Dans une base révisée minimale, le maximum d'assertions de la clôture déductive de la base est conservé, c'est-à-dire que pour un ensemble B'_-, l'ensemble B'_+ est maximal selon l'inclusion ensembliste.

2. Interprétation "objet" de la minimalité

Les définitions et propositions précédentes précisent ce qu'est la notion de minimalité de manière théorique, mais quelles sont les assertions qui appartiennent aux bases contractées minimales ? La proposition suivante donne une caractérisation syntaxique des bases contractées minimales, en indiquant les assertions qui sont conservées en fonction de la structure de la base et en particulier de la relation de spécialisation.

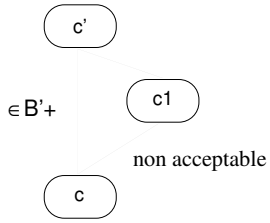
PROPOSITION 23 : INTERPRÉTATION "OBJET" DE LA MINIMALITÉ

B' est une base contractée minimale de (B,δ) ssi :

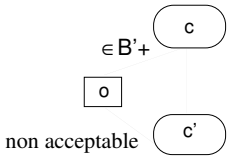
(0) B' est une base contractée de (B,δ).

(1) $\forall\ B''$ base contractée de (B,δ), si $B''_{-} \subseteq B'_{-}$ et $B'_{+} \subseteq B''_{+}$, alors $B'=B''$.

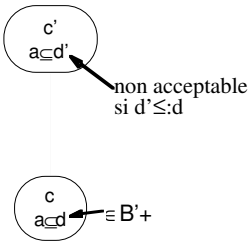
(2) $\forall (c \leq c') \in B'_{+}$, $\forall c_1 \neq c'$, si $B' \vdash c_1 \leq c'$ et $(c \leq c_1) \in \mathrm{Cn}(B)$, alors $(c \leq c_1)$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.



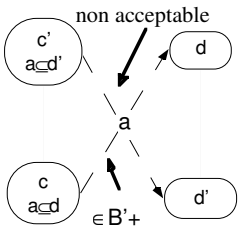
(3) $\forall (o \in c) \in B'_{+}$, $\forall c' \neq c$, si $B' \vdash c' \leq c$ et $(o \in c') \in \mathrm{Cn}(B)$, alors $(o \in c')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.



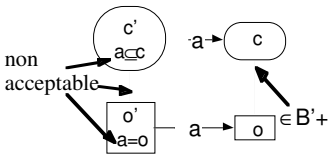
(4) $\forall\ (c.a \subseteq d) \in B'_{+}$, $\forall (c',d') \neq (c,d)$, si $d' \leq : d$, $B' \vdash c \leq c'$ et $(c'.a \subseteq d') \in \mathrm{Cn}(B)$, alors $(c'.a \subseteq d')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.



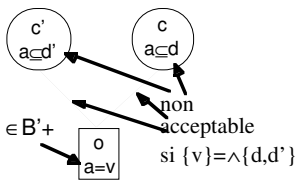
(5) $\forall\ (c.a \subseteq d) \in B'_{+}$, $\forall (c',d') \neq (c,d)$, $d,d' \in C$, si $B' \vdash d' \leq d$, $B' \vdash c \leq c'$ et $(c'.a \subseteq d') \in \mathrm{Cn}(B)$, alors $(c'.a \subseteq d')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.



(6) $\forall (o \in c) \in B'_{+}$, $\forall c' \neq c$, o', a si $(c'.a \subseteq c) \in \mathrm{Cn}(B)$, $(o' \in c') \in \mathrm{Cn}(B)$ et $(o'.a=o) \in \mathrm{Cn}(B)$, alors l'une de ces trois assertions n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.



(7) $\forall (o.a=v) \in B'_{+}$, pour tout ensemble d'assertions $\{c_i.a \subseteq d_{ij}, o \in c_i\} \in \mathrm{Cn}(B)$ minimal tel que $\{v\} = \wedge d_{ij}$, alors l'une des assertions n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.



Cette proposition indique qu'une base est une base contractée minimale si :

(0) la base est une base contractée,

(1) aucune autre base contractée ne fait moins de suppressions et plus d'ajouts,

(2) toute classe est sous-classe de ses anciennes super-classes les plus spécifiques possibles[17],

(3) toute instance est instance de la classe la plus spécifique possible,

(4) toute restriction d'attribut par un domaine est effectuée dans la classe la moins spécifique possible et avec un domaine contenant la restriction d'attribut initiale et le plus réduit possible,

(5) propriété similaire à la précédente pour les restrictions d'attribut de classe par des classes : toute restriction d'attribut est effectuée dans la classe la moins spécifique possible et avec une classe, super-classe de la classe initiale, la plus spécifique possible,

(6) une assertion du type ($o \in c$) n'est ajoutée explicitement dans la base contractée que si elle n'est pas déductible, pour assurer que la base contractée est sous forme première,

(7) propriété similaire à la précédente pour les assertions du type ($o.a=v$).

Elle est importante en particulier pour pouvoir déterminer de manière effective si les solutions proposées par un algorithme sont minimales.

EXEMPLE :

Dans l'exemple de base inconsistante illustré dans la Figure 34, on a donc :

$B_1 \propto B_2$ (d'après le troisième point de la proposition),

$B_3 \propto B_4$ (d'après le second point de la proposition) et

$B_6 \propto B_5$, (d'après les premier et second points de la proposition).

Les autres bases révisées ne sont pas comparables.

Ainsi, la notion de base contractée minimale s'exprime syntaxiquement et sémantiquement. Dès lors, les procédures de révision qui peuvent être mises en œuvre syntaxiquement auront le sens de «conserver le maximum d'assertions valides».

D. Principes satisfaits

Nous allons étudier comment se situe notre définition de la révision par rapport aux différents principes énoncés dans le paragraphe I.F.1.b. Afin de pouvoir comparer la révision

[17] Par "classe la plus spécifique possible", on entend la classe la plus spécifique telle que l'assertion était déductible de la base initiale et est acceptable dans la base contractée.

et la minimalité proposées avec d'autres travaux, nous supposons que le concepteur choisit parmi les différentes solutions proposées. La base révisée obtenue dans la révision en logique sera donc comparée à une base révisée quelconque qui satisfait les critères de minimalité présentés. Nous mettons l'accent pour chaque point sur la comparaison avec les travaux de Alchourròn, Gärdenfors et Makinson, de Nebel et de Hansson. La section suivante est consacrée à la synthèse des principales originalités de notre approche.

1. Représentation de la base révisée

La base révisée est exprimée comme la base initiale sous forme première, ce qui permet d'effectuer des ajouts successifs. De plus, les nouvelles assertions sont de l'un des six types possibles. Les ensembles d'entités (O, A, C) ne sont pourtant pas fixés au départ, mais évoluent au cours des ajouts d'assertions. Ainsi, l'ajout de ($o \in c$) nécessite l'ajout de o dans l'ensemble O si l'instance n'y appartenait pas déjà. Par contre, les ensembles V et T sont définis indépendamment des assertions ajoutées et donc ($o.a=v$) n'est pas une assertion consistante si v n'est pas un élément de V. Les bases ne sont pas exprimées sous forme de clôture déductive donc (G1) et (N3) ne sont pas satisfaits, par contre (N1) et (D1) le sont.

2. Ajout de la dernière connaissance

La dernière assertion est ajoutée dans la base, à condition qu'elle soit différente de ($c.a \subseteq \perp$) ; c'est en effet le seul cas où l'assertion est incohérente en elle-même. (G2) est satisfait dans notre cas si on considère des assertions cohérentes en elles-mêmes ; il n'existe en effet pas d'assertion qui soit inconsistante en elle-même, en particulier car il n'y a pas d'opérateur $\wedge$ entre les assertions. De même (K1), (D4) et (N4) ne sont pas satisfaits dans le cas général mais uniquement si l'assertion n'est pas incohérente en elle-même, ce qui est exprimé par le postulat (H-S3).

Toute dernière assertion cohérente est donc prise en compte lors de la révision. Cependant, nous verrons que ceci n'est pas toujours souhaitable et que le concepteur peut souhaiter ne pas tenir compte de son assertion ou la modifier (cf. section IV.D).

3. Consistance de la base révisée

Les postulats (G5), (K3), (D3) et (H-C4) sont satisfaits. De plus, la base révisée est toujours acceptable, c'est-à-dire qu'elle est non seulement consistante, mais également cohérente.

4. Préservation de la connaissance et maintien des justifications

Ces principes nécessitent un peu plus de développements que les précédents et s'organise en quatre points détaillés dans la suite :

a) Conservation du maximum d'assertions, selon l'inclusion ensembliste, de la clôture déductive ;
b) Violation du principe de maintien des justifications par la révision proposée ;
c) Utilisation combinée des deux principes pour une révision plus naturelle ;
d) Comparaison avec les postulats énoncés dans la littérature à ce propos, ce qui nécessite d'étendre la définition de la révision au cas où l'assertion est acceptable.

a) Conservation du maximum d'assertions

La notion de minimalité proposée dans ce travail peut être caractérisée de manière informelle par la préservation du maximum d'assertions qui étaient déductibles de la base initiale. Cette notion de "maximum" se base sur l'inclusion ensembliste et non sur le nombre d'assertions séparant les clôtures déductives initiale et finale et ne permet pas de définir *une* plus petite base révisée. Il existe différents types de solutions, selon les assertions qui sont supprimées (Figure 36).

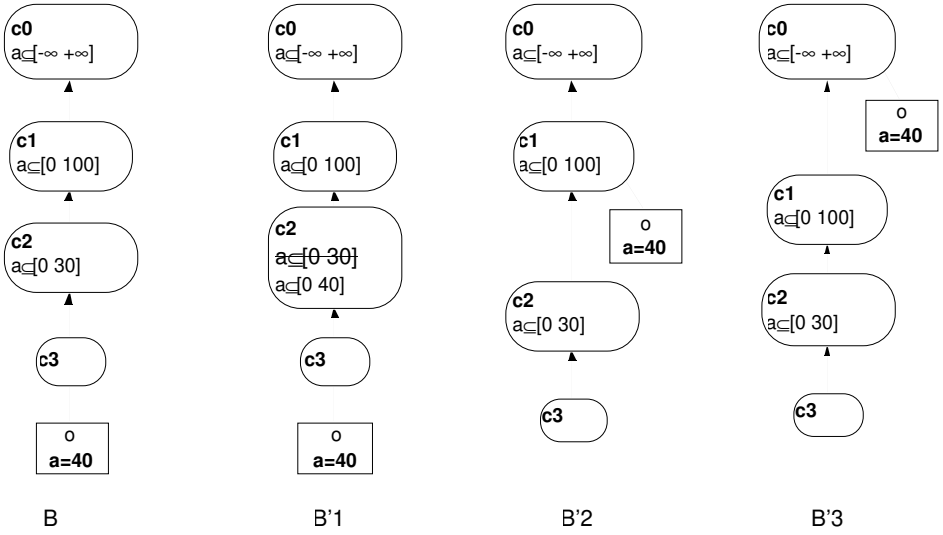


Figure 36 : Dans la base B, l'assertion ($o.a$=40) soulève une inconsistance ; parmi les bases révisées possibles, trois sont illustrées. Dans Cn($B'1$), un nombre important d'assertions de Cn(B) n'apparaissent pas : toutes les assertions s'exprimant ($c_2.a \subseteq d$) et ($c_3.a \subseteq d$) avec d intervalle contenant [0 30] et contenu dans [0 40] ; en effet, comme $c_2.a \subseteq$[0 30] est remplacé par $c_2.a \subseteq$[0 40], il n'est plus possible de déduire $c_2.a \subseteq$[0 31], $c_2.a \subseteq$[0 32], etc. Dans Cn(B'_2), il ne manque que deux assertions (($o \in c_3$) et ($o \in c_2$)) par rapport à Cn(B) et dans Cn(B'_3), il en manque trois (($o \in c_1$) en plus des précédentes). Il ne fait pas de doute que la troisième solution n'est pas minimale car Cn(B'_2) contient Cn(B'_3) ; B'_2 au contraire est une base révisée minimale. Quant à la première solution, elle fait un grand nombre de suppressions, mais il n'existe pas de bases révisées qui la contienne. Elle est donc considérée comme minimale, même si elle fait plus de modifications en nombre que B'_2 par exemple.

Nous avons considéré que chaque type de solution est intéressante et donc que le nombre d'assertions différentes entre les clôtures déductives (ou les formes premières) initiale et finale n'est pas un critère de choix pertinent.

b) Violation du principe de maintien de justifications

La préservation du maximum de connaissances se fait au détriment du maintien des justifications qui consisterait à supprimer les assertions qui ne sont que des conséquences d'assertions supprimées.

Un exemple simple permet de voir la différence entre les deux approches : supposons qu'une valeur d'attribut d'une instance est déduite par la règle de (o-valuation) car la restriction d'attribut de l'une de ses classes d'appartenance est un singleton ; si la description d'attribut de la classe est modifiée dans la base révisée, cette valeur d'attribut d'instance est tout de même préservée.

B= $\{c_1 \leq c_0 ; i \in c_1 ; c_0.a \subseteq [10\ 30] ; c_1.a \subseteq [30\ 100]\}$

FP(B)= $\{c_1 \leq c_0 ; i \in c_1 ; c_0.a \subseteq [10\ 30] ; c_1.a \subseteq [30\ 30]\}$

Cn(B)= $\{c_1 \leq c_0 ; i \in c_1 ; c_0.a \subseteq [10\ 30] ; c_1.a \subseteq [30\ 100] ; c_1.a \subseteq [30\ 30] ; i.a = 30\}$

Si la révision consiste à supprimer l'assertion $c_1.a \subseteq [30\ 30]$, une base révisée minimale correspondante sera $B' = \{c_1 \leq c_0 ; i \in c_1 ; c_0.a \subseteq [10\ 30] ; i.a = 30\}$, alors que $c_1.a \subseteq [30\ 30]$ n'est plus déductible. Ce choix est fondé sur l'hypothèse suivante : le concepteur, en donnant explicitement certaines assertions, approuve également toutes celles qui pourront en être déduites et donc lors de la révision, il souhaite perdre le moins possible d'assertions parmi toutes celles qui étaient auparavant déductibles de la base.

En effet, les assertions déduites sont généralement des assertions aux contraintes relâchées sur les instances : ainsi si une instance est rattachée à une classe, le concepteur approuve l'appartenance de l'instance à toutes les super-classes, et elle est donc déplacée sous une super-classe. Les assertions déduites reflètent généralement ce que le concepteur voulait mettre dans la base, et pas seulement ce qui a pu être tiré des connaissances qu'il a fournies. En effet, la plupart des règles d'inférence permettent de déduire des assertions à partir d'assertions qui y sont directement liées : ($\leq$-transitivité), ($\in$-clôture), (d-intersection), (d-héritage), (c-héritage), (d-clôture) et (c-clôture) sont des règles qui permettent de déduire des assertions moins contraintes que les assertions qui ont permis de les déduire. Lorsqu'une assertion est supprimée dans une base contractée, il est naturel de vouloir conserver une assertion similaire qui soit la plus contrainte possible. Ainsi, si on supprime l'attachement d'une instance à une classe, cette instance est rattachée à la super-classe la plus spécifique ; de même, les restrictions d'attribut de classe sont remplacées par des restrictions moins contraignantes.

Cependant, le problème est plus crucial lorsqu'il s'agit de faire des ajouts d'assertions qui ont été déduites par la règle de (o-valuation) ou de ($\in$-inférence). La règle de (o-valuation) attribue automatiquement une valeur à un attribut d'une instance s'il n'y a qu'une valeur possible. La valeur en question n'a pas forcément beaucoup de signification pour le concepteur et la conserver dans les bases révisées si les assertions permettant de la déduire ont été supprimées est alors sûrement surprenante. Le même problème se pose pour les rattachements d'instance à des classes par la règle de ($\in$-inférence).

c) Combinaison du maintien des justifications et de la préservation du maximum d'assertions

Dans l'algorithme, il est donc proposé au concepteur non seulement les bases révisées minimales, mais également des bases qui font une entorse à la notion de minimalité, c'est-à-dire dans lesquelles les assertions précédemment déduites par les règles de (o-valuation) et de ($\in$-inférence) ne sont pas préservées.

Pour caractériser de telles bases, il est possible de définir la clôture sans tenir compte de ces deux règles, ce qui permet d'appliquer le principe de maintien de justification pour les assertions du type ($o \in c$) ou ($o.a=v$) déduites respectivement par ($\in$-inférence) et (o-valuation) et le principe de préservation de maximum de connaissances pour les autres assertions. Le problème est alors que l'équivalent sémantique de la minimalité définie de manière syntaxique est moins immédiat.

d) Adaptation de notre notion de révision aux opérateurs de la littérature

Afin de comparer la révision proposée ici et les postulats qui sont donnés dans la littérature, il faut quelque peu étendre sa définition. En particulier, si δ est consistante avec la base, ce qui peut se voir comme $(\neg\delta) \notin B$ dans le formalisme logique avec B une base, la révision n'est pas définie, car nous avons choisi de n'appeler révision que l'ajout d'une assertion inconsistante ou incohérente avec la base initiale. Cependant, la définition peut être étendue en définissant la révision par une assertion cohérente, comme la forme première de l'union avec la base existante. Dans ce cas, (G4) est satisfait. (P3) est satisfait car il se traduit dans notre formalisme par le fait que si l'assertion ajoutée n'est pas consistante avec la base, les assertions supprimées sont celles qui ne sont pas déductibles. (N5) et (D5) sont satisfaits, la minimalité correspondant à la minimalité de la différence ensembliste entre clôtures déductives de la base initiale et de la base révisée.

5. Non pertinence de la syntaxe

La pertinence ou non de la syntaxe n'est pas aussi cruciale qu'en logique, en particulier par l'absence de connecteurs ; de ce fait, bien peu de formulations différentes sont équivalentes. Le seul cas vient des restrictions d'attributs par les domaines. Cependant, comme la révision

se fonde sur la forme première, la syntaxe n'est pas prise en compte, puisque deux bases équivalentes, c'est-à-dire ayant la même clôture déductive, ont la même forme première (Proposition 11). (G6), (K4), (D2) et (N2) sont satisfaits, contrairement à ce qui est proposé dans [Winslett86] et [Fagin83].

6. Confinement

Vu la définition d'une base révisée, aucune connaissance nouvelle, c'est-à-dire n'appartenant pas à la clôture déductive initiale, n'est ajoutée, sauf évidemment les déductions suite à l'ajout de la dernière assertion. (G3) exprime que la révision n'introduit pas de connaissances nouvelles. Ce postulat est plus strict dans notre cadre que dans le cadre logique car la clôture déductive en cas d'inconsistance ne contient pas toutes les formules du langage. Il est cependant vérifié, car la clôture déductive de la base révisée est inclue dans la clôture déductive de l'union de la base et de l'assertion ajoutée ($\mathrm{Cn}(B' \cup \{\delta\}) \subset \mathrm{Cn}(B \cup \{\delta\})$). (K2) et (H-R2) sont satisfaits.

7. Équité

Le choix entre les différentes bases révisées minimales est laissé au concepteur, car les connaissances contenues dans la base ne permettent pas de décider quelle est la meilleure. En effet, une base admet plusieurs modèles, dont l'un est conforme avec le domaine qu'il veut décrire et ce modèle particulier n'est pas connu. Parmi les bases révisées, il y en a donc une qui permet de décrire au mieux le domaine, mais il n'est pas possible de la repérer automatiquement, car les seules informations disponibles sont celles contenues dans la base, ce qui se révèle insuffisant. De même, on ne peut imaginer faire une intersection de toutes les solutions possibles, car chacune correspond à une interprétation bien particulière et que cela reviendrait donc à perdre beaucoup trop de connaissances. Évidemment, cette intersection représente la partie de la base qui n'est pas touchée par l'inconsistance et qui donc n'est pas remise en cause, ce qui est intéressant à connaître et éventuellement à transmettre au concepteur. Cependant, l'intersection de toutes les bases révisées possibles ne nous semble pas acceptable, car elle n'est pas minimale.

Il ne reste donc qu'à espérer que le choix du concepteur parmi les différentes solutions qui lui seront proposées ne sera pas arbitraire, et conduira à une base qui sera le plus proche possible de sa représentation du domaine à modéliser. Par contre, l'ensemble de bases révisées qui lui est proposé est indépendant de tout contexte, afin qu'il puisse toujours trouver celle qui lui convient s'il recherche effectivement une solution qui fasse supprimer le moins possible de connaissances de la base.

Prenons l'exemple suivant :

Soit une base $B=\{c.a\subseteq[12\ 20],\ o\in c\}$ et l'assertion $o.a=11$. Les deux bases révisées minimales possibles sont : $B1=\{c.a\subseteq[11\ 20],\ o\in c,\ o.a=11\}$ et $B2=\{c.a\subseteq[12\ 20], o.a=11\}$.

Considérons deux des interprétations I1 et I2 de la base.

I1(c)={étudiants admis}	I2(c)={jeunes}
I1(o)= Pierre	I2(o)= Jean
I1(a)= étudiant → moyenne annuelle	I2(a)= personne → âge

Dans le premier cas, Pierre, étant le petit-fils du directeur de l'école, ne peut pas redoubler son année, et comme les notes ont été communiquées, la barre d'admissibilité est changée : $c.a\subseteq[12\ 20]$ est remplacée par $c.a\subseteq[11\ 20]$. Dans le second cas, le concepteur souhaite conserver sa définition de la classe des jeunes et Jean n'est donc pas considéré comme un jeune : $(o\in c)$ est supprimée.

(D6) est donc satisfait dans la mesure où le choix du concepteur n'est pas arbitraire.

8. Complexité

Les mécanismes de révision ont été peu développés jusqu'à présent dans la littérature, pour des raisons de complexité. En effet, tous les moyens de réviser une base de connaissance ont besoin d'un oracle leur disant si la base est consistante ou non. Si la complexité de cette décision est exponentielle, la révision est véritablement impraticable [Nebel90].

Dans le cas présenté ci-dessus, cette déduction est polynomiale dans le pire des cas (voir l'analyse de la complexité en annexes C et D) et l'expérience pratique montre que les systèmes à objets permettent de décider rapidement de la consistance du système, du moins ceux relativement simples, c'est-à-dire sans problème de satisfaction de contraintes, ni concepts récursifs ou restriction de rôles ouverts. En effet, nous montrons que les algorithmes pour obtenir *une* base révisée sont en coût polynomial de la taille de la base dans le cas du langage simple et du langage complexe. Par contre, le nombre de solutions est, pour l'ajout de certaines assertions précisées lors du calcul de la complexité, exponentiel en fonction du nombre de classes, d'objets ou d'attributs ; l'obtention de *l'ensemble* des bases révisées est alors de coût exponentiel. Nous étudions en détail la complexité en annexes C et D.

9. Synthèse pour quelques travaux

a) Les postulats donnés par AGM

PROPOSITION 24 : SATISFACTION DES POSTULATS (AGM)

Les postulats (AGM) (G2-6) sont satisfaits par la révision proposée au chapitre III.

Seul le postulat (G1) n'est pas satisfait car les bases révisées ne sont pas sous forme de clôture déductive, mais sous forme première.

b) La contraction maxi-choix

La minimalité proposée dans la section III.C consiste à obtenir une base dont la clôture déductive est un sous-ensemble maximalement consistant de la clôture de la base initiale, du point de vue de l'inclusion ensembliste. Ceci s'apparente donc à la révision basée sur la contraction maxi-choix de [Alchourròn&85], en laissant le choix entre les différentes bases possibles au concepteur, c'est-à-dire que la fonction de choix *gamma* est le choix du concepteur. Évidemment, cette analogie n'est possible que grâce au système déductif particulier proposé pour les langages à objets, et c'est ce système déductif qui confère sa particularité à la révision proposée.

L'une des principales critiques ([Nebel90, paragraphe 6.2.3]) de cette opération est qu'elle conduit à une théorie fermée, c'est-à-dire une théorie telle que pour toute formule *f* du langage, il est possible de déduire *f* ou $\neg f$. Ceci est problématique car l'ajout d'une connaissance permet de déduire des connaissances qui en sont complètement indépendantes. Cet inconvénient n'apparaît pas un dans notre cadre, car la négation n'est pas définie.

c) Révision sur des bases finies

La révision ne fait pas de distinction entre les faits donnés et les faits déduits car elle est définie par le biais de la forme première. Cependant, dans la forme première, il y a principalement les assertions données explicitement par le concepteur, à l'exception des restrictions d'attributs de classes qui sont calculées. Le rôle de la forme première est plutôt de supprimer les assertions redondantes car déductibles d'autres assertions. Donc, on peut considérer que la révision proposée agit sur une base de faits donnés, sachant que la minimalité s'évalue sur les bases fermées par déduction mais que cela se ramène à un ordre sur les modifications effectués sur la forme première.

Contrairement à [Nebel92], il n'existe pas d'ordre entre les assertions. Une des perspectives de ce travail consiste à introduire des critères permettant d'ordonner partiellement la connaissance.

d) Critères de Hansson

La révision définie dans notre travail satisfait un des ensembles de contraintes maximaux (S3, C4, P3, R2) des postulats de [Hansson96], c'est-à-dire tels qu'il n'existe pas d'opérateur de révision permettant de satisfaire un postulat plus strict pour l'un des critères, les autres restant inchangés.

E. Synthèse des principales originalités et caractéristiques de la révision des langages d'objets

La Figure 37 illustre le processus de révision et ses principales caractéristiques, à savoir :

- Grâce à la définition syntaxique de l'inconsistance et de l'incohérence, il est possible de décider si une assertion est acceptable dans une base ou non ; l'algorithme et sa complexité sont donnés en annexes C et D.
- Les bases recherchées sont *acceptables*, c'est-à-dire consistantes, ce qui correspond à la définition classique, *et* cohérentes (toute classe peut contenir des instances).
- Les assertions *acceptables* sont *simplement ajoutées* dans la base.
- La révision est définie uniquement quand l'assertion est inacceptable dans la base et cohérente en elle-même ; la seule assertion incohérente est du type $c.a \subseteq \bot$.
- Le *choix* parmi les différentes bases révisées minimales *est laissé au concepteur*, afin que la base révisée corresponde au mieux au domaine qu'il veut décrire. La révision proposée ne fournit pas une seule base révisée, contrairement à la plupart des définitions des opérateurs de révision sous forme de fonction. Cette définition sous forme de fonction de la base et de l'assertion introduite est d'ailleurs remise en cause par [Friedman&96] qui stipule que ceci n'est possible que si le langage est suffisamment expressif pour traduire la préférence.

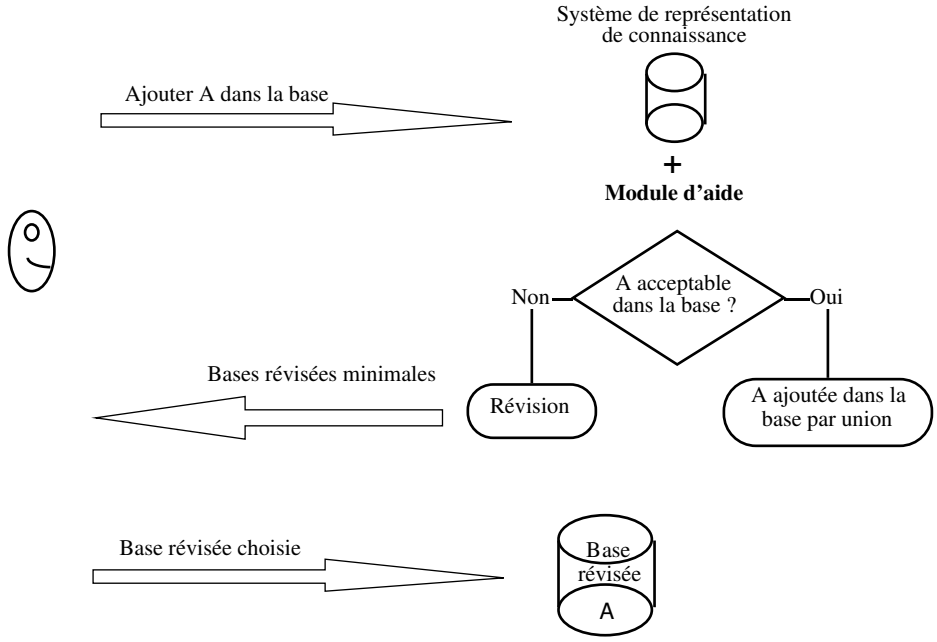


Figure 37 : Résumé du processus de révision de la base

Enfin, grâce au mécanisme de déduction proposé, la *minimalité*, définie par rapport à l'inclusion ensembliste, *s'interprète de manière intuitive* :

- les assertions ne sont pas supprimées inutilement de la forme première,
- les instances (resp. les classes) sont rattachées à la classe (resp. déplacées sous leur super-classe) la plus spécifique,
- les domaines sont agrandis au minimum.

La révision proposée s'adapte donc aux représentations de connaissance par objets – en définissant en particulier une minimalité qui s'interprète intuitivement par rapport à la relation de spécialisation notamment – et qui peut être implémentée par un algorithme dont la complexité est polynomiale dans le pire cas. De plus, elle satisfait un ensemble significatif des principaux postulats posés en logique.

Il reste à présent à montrer comment la révision proposée peut être implémentée efficacement dans un système de représentation de connaissance par objets.

Chapitre IV. Algorithme et expérimentation

Nous définissons dans ce chapitre l'algorithme permettant de déterminer les bases révisées. L'objectif est d'obtenir les bases révisées sous la forme FP(B')=FP(B)- $B'_- \cup B'_+$, ce qui se résume donc à déterminer B'_- et B'_+. Cette écriture de base révisée est explicitée sous forme algorithmique si nécessaire. La section A donne quelques généralités sur l'algorithme; l'algorithme pour le langage simple (resp. le langage complexe) est donné en section B (resp. section C). Obtenir une base révisée minimale ne correspond cependant pas toujours à ce que le concepteur souhaite faire; c'est pourquoi la section D étudie d'une part une alternative à la révision qui consiste à modifier l'assertion qui provoque l'inconsistance et d'autre part, des cas où des connaissances de la clôture déductive sont supprimées car les assertions ayant permis de les déduire ont été supprimées.

D'autre part, nous étudions l'application, au système de représentation de connaissance par objets TROPES, de la représentation formelle d'un langage d'objets et de la révision (section E). Nous donnons enfin des exemples d'exécution du programme correspondant à l'algorithme (section F), ainsi que quelques fonctionnalités supplémentaires qui complètent l'algorithme pour que la révision soit plus facile à utiliser par le concepteur de la base (section G).

A. Généralités sur l'algorithme

Nous avons jusqu'à présent défini formellement ce qu'est une base révisée; il s'agit maintenant de décrire un algorithme permettant d'obtenir ces bases révisées.

1. Description de l'algorithme

Pour pouvoir être mis en œuvre concrètement, l'algorithme se base sur la forme première car c'est un ensemble fini et minimal d'assertions qui permet de déduire l'ensemble de la base.

Par abus de langage, on appellera inconsistance un motif d'inconsistance et incohérence une classe incohérente.

Rappelons que la révision d'une base s'effectue toujours pour une seule assertion à la fois, mais que plusieurs inconsistances ou incohérences peuvent évidemment survenir lors de l'ajout d'une assertion. Cet algorithme est fonction de l'assertion qui a provoqué

l'inconsistance ou l'incohérence, ou, plus justement, de la dernière assertion ajoutée à la base avant la détection d'un problème. En effet, cette assertion doit appartenir à la base révisée par définition des bases révisées, même si nous verrons que pour enrichir ce mécanisme de révision, il est proposé au concepteur de modifier son assertion (section D). Les solutions proposées sont également liées au type de problème (incohérence ou l'un des trois types d'inconsistance). Plusieurs solutions sont possibles pour résoudre un type de problème particulier indépendamment de la dernière assertion ajoutée mais il ne faut tenir compte que de celles qui ne remettent pas en cause cette dernière assertion.

L'algorithme est donc décrit pour chacun des quatre types d'assertion. Pour chacun, il sera donné :

- l'ensemble des incohérences et/ou inconsistances qui peuvent apparaître suite à l'ajout de cette assertion;
- les différentes solutions possibles en définissant les ensembles B'_- et B'_+ ; ces solutions peuvent être partielles, c'est-à-dire ne résoudre qu'une partie des problèmes soulevés;
- la solution globale combinant les différentes solutions élémentaires.

Toutes les solutions proposées doivent être minimales quel que soit le type de l'assertion; cette minimalité est prouvée grâce à la proposition sur l'interprétation objet de la minimalité (Proposition 23 page 105). La preuve est donnée en annexe.

La description de l'algorithme est relativement complexe car il existe de nombreuses règles d'inférence et que l'on tient compte de la forme des assertions lors de la description des solutions. L'objectif est également de tirer parti au mieux de la structure de la base afin d'éviter toute énumération inutile, d'obtenir ainsi un algorithme efficace et de présenter clairement le déroulement du processus de révision au concepteur de la base. Notamment la minimalité est atteinte en se focalisant sur les assertions qui ne sont plus déductibles suite aux suppressions d'assertions.

2. Hypothèses supplémentaires

Cet algorithme est celui qui a été utilisé pour réaliser l'implémentation au-dessus de TROPES, l'hypothèse supplémentaire de mono-spécialisation est donc posée. D'autre part, les instance ne sont attachées qu'à une seule classe par point de vue (voir la description en I.C.2). Cette hypothèse se traduit par :

Si $(o \in c) \in \mathrm{FP}(B)$ et $(o \in c') \in \mathrm{FP}(B)$ avec $c \neq c'$, alors il n'existe pas de classe c'' telle que $(c \leq^* c'') \in \mathrm{FP}(B)$ et $(c' \leq^* c'') \in \mathrm{FP}(B)$.

De plus, on admet que la racine de chaque hiérarchie de classes ne pose aucune restriction sur les domaines de ses attributs.

3. Rappel sur la forme première

Avec les hypothèses supplémentaires posées, la forme première (Définition 12 page 70) est formée des assertions suivantes :

- au plus une valeur pour un attribut d'instance si celle-ci n'est pas déductible par la règle de (o-valuation);
- pour chaque classe (sauf pour les classes qui sont racines de hiérarchies), une assertion indiquant sa relation de spécialisation avec sa super-classe;
- pour chaque instance, ses classes de rattachement les plus spécifiques;
- au plus une restriction pour chaque attribut de classe; de plus, les restrictions d'attribut sont ordonnées par l'inclusion ensembliste le long de chaque branche; la restriction de l'attribut dans une sous-classe est inclue dans tous les domaines de cet attribut dans ses super-classes.

4. Notations

Nous regrouperons sous le terme *problème* les notions d'inconsistance et d'incohérence.

Pour que l'expression de l'algorithme soit plus compacte et plus claire, nous définissons un certain nombre de notations ci-dessous :

Pour toute classe c,

sup(c) dénote sa super-classe directe (qui est unique grâce à l'hypothèse de mono-spécialisation) : $(c \leq \text{sup}(c)) \in \text{FP}(B)$;

sous(c,c') dénote sa sous-classe directe qui est aussi super-classe de c' : $(\text{sous}(c,c') \leq c) \in \text{FP}(B)$ et $(c' \leq^* \text{sous}(c,c')) \in \text{FP}(B)$;

racine(c) dénote la classe racine de sa hiérarchie : $(c \leq^* \text{racine}(c)) \in \text{FP}(B)$ et il n'existe pas $c' \neq \text{racine}(c)$ t.q. $(\text{racine}(c) \leq^* c') \in \text{FP}(B)$.

$[c\ c']$ dénote l'ensemble de classes qui sont super-classe de c et sous-classe de c' :

$$[c\ c'] = \{c'' \text{ t.q. } (c \leq^* c'') \in \text{FP}(B) \text{ et } (c'' \leq^* c') \in \text{FP}(B)\}$$

Les notations $]c\ c']$, $[c\ c'[$ et $]c\ c'[$ sont les extensions naturelles de cette notation dénotant respectivement l'ensemble $[c\ c']$ moins la classe c, moins la classe c' et moins les deux.

Nous aurons besoin de savoir si une assertion du type $(o.a=v)$ est encore déductible après certaines suppressions, c'est pourquoi nous introduisons Rest(o,a,Cl) qui est le domaine des

valeurs possibles pour l'attribut a de l'instance o si on ne tient compte que des classes d'appartenance qui n'appartiennent pas à Cl.

$$\text{Rest}(o,a,Cl)= \bigwedge_j \{d_j \text{ t.q. } (o\in c_j)\in \text{Cn}(B), (c_j.a\subseteq d_j)\in \text{FP}(B) \text{ et } c_j\notin Cl\}$$

La plus spécifique super-classe commune à c et c' est dénotée PSSCC(c,c') :

PSSCC(c,c')=c'' t.q. $(c\leq^* c'')\in \text{FP}(B)$, $(c'\leq^* c'')\in \text{FP}(B)$ et il n'existe pas $c'''\neq c''$ t.q. $(c'''\leq^* c'')\in \text{FP}(B)$, $(c\leq^* c''')\in \text{FP}(B)$ et $(c'\leq^* c''')\in \text{FP}(B)$

Dans la définition des bases révisées, il sera utile de savoir localiser d'où vient le problème et en particulier de connaître la classe dans laquelle est posée la restriction d'attribut problématique; en fait, il s'agit de la classe la plus spécifique ayant une définition de restriction d'attribut, vu la définition de la forme première. En effet, si le domaine qui restreint un attribut dans une classe ne peut contenir un domaine donné, il en sera de même de tous les domaines des sous-classes. Inversement si un domaine qui restreint un attribut peut contenir un domaine quelconque, ce domaine sera aussi inclus dans les restrictions d'attribut de toutes les super-classes. Cette classe qui contient la première restriction problématique est définie formellement ci-dessous :

DÉFINITION 31 : PREMIÈRE CLASSE INCONSISTANTE (AVEC UNE RESTRICTION PAR UN DOMAINE)

PCI(c,a,d) est la classe la plus spécifique qui est super-classe de c et qui n'est pas compatible avec le domaine d pour l'attribut a :

PCI(c,a,d)= c' t.q. $(c\leq^* c')\in \text{FP}(B)$, $(c'.a\subseteq d')\in \text{FP}(B)$ et $\neg d\leq: d'$.

Remarquons que lorsque l'on est sûr qu'il existe une inconsistance, vérifier $\neg d\leq: d'$. n'est pas nécessaire, il suffit de trouver la super-classe la plus spécifique qui admet une restriction d'attribut dans la forme première.

Il s'agit également de détecter la classe dont toutes les super-classes sont compatibles avec une nouvelle restriction pour un attribut donné : celle qui admet une restriction d'attribut qui contient le nouveau domaine et qui est appelée la première classe possible.

DÉFINITION 32 : PREMIÈRE CLASSE POSSIBLE

La première classe possible d'une classe c pour un attribut a et un domaine d est la classe c_m qui est la super-classe de c la plus spécifique compatible avec le domaine d pour l'attribut a, notée PCP(c,a,d), c'est-à-dire :

PCP(c,a,d)= c_m t.q.

pour tout c_i telle que $(c_m\leq^* c_i)\in \text{FP}(B)$ et $(c_i.a\subseteq d_i)\in \text{FP}(B)$ alors $d\leq: d_i$.

et si $(\text{sous}(c_m,c).a\subseteq d')\in \text{FP}(B)$ alors $\neg d\leq: d'$.

B. Algorithme sur le langage simple

Afin de mieux expliquer les stratégies de révision, nous commençons par énumérer les modifications possibles de la forme première de la base. Comme les bases comportent des assertions de quatre type, les modifications élémentaires correspondent à la suppression de chacun de ces quatre types d'assertions. Cette suppression est complétée par l'ajout des assertions permettant d'avoir une base révisée conservant le maximum d'assertions de la clôture déductive de la base initiale.

Lors de la révision, les modifications possibles de la base sont le déplacement d'une instance ou d'une classe, la suppression de la valeur d'un attribut d'une instance ou la modification de la restriction d'un attribut de classe. Les solutions élémentaires sont donc :

- supprimer une valeur d'attribut d'une instance,
- déplacer une instance sous l'une de ses super-classes, ce qui revient à supprimer son attachement à une classe actuelle,
- supprimer le lien de spécialisation d'une classe avec sa super-classe et en créer un avec l'une de ses super-classes, ce qui revient à déplacer une classe sous l'une de ses sous-classes,
- supprimer la restriction d'attribut d'une classe et la remplacer par une restriction plus faible, ce qui revient à agrandir les domaines qui restreignent les attributs de classes.

1. Modifications élémentaires de la base sur la forme première

Les modifications élémentaires sont définies par la donnée de B'_- et B'_+ et le nom qui leur est donné va être utilisé dans la définition des bases révisées. De plus, la pré-condition éventuelle d'utilisation de ces modifications est donnée.

a) Supprimer une valeur d'un attribut d'une instance (S)

Il s'agit de supprimer l'assertion ($o.a=v$) de la forme première.

S(o,a,v) : $B'_-=\{o.a=v\}$ et $B'_+=\emptyset$

b) Déplacer une instance dans une super-classe (DI)

Cela consiste à couper le rattachement de l'instance avec sa classe actuelle et à l'attacher à l'une des super-classes si cet attachement n'est pas déjà déductible.

De plus, afin de préserver le maximum de connaissance de la clôture déductive, il faut préserver toutes les assertions du type ($o.a=v$) qui ne peuvent plus être déduites suite au déplacement de l'instance (Figure 38). Ce sont les assertions qui étaient déductibles par (o-

valuation), ce qui s'exprime par $\{v\}$=Rest($o,a,\emptyset$), et qui ne le sont plus après déplacement à cause des restrictions perdues (les restrictions des classes [$c\ c'$[).

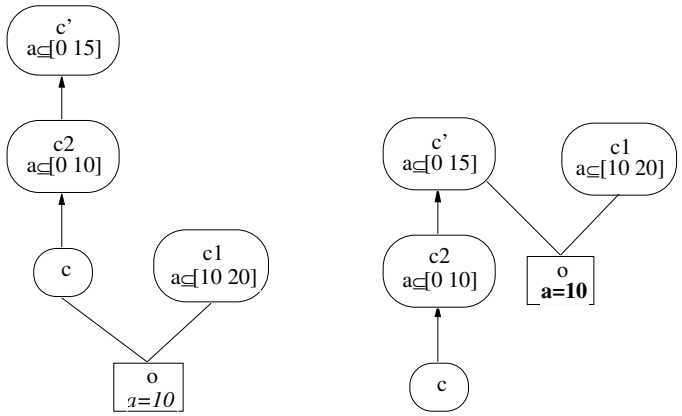


Figure 38 : Lors du déplacement de l'instance o de la classe c à la classe c', l'assertion ($o.a$=10) ne peut plus être déduite, donc elle est explicitement ajoutée dans la base révisée.

La modification élémentaire de déplacement d'instance s'exprime donc formellement par :

pré-condition : $(c \leq^* c') \in \mathrm{FP}(B)$

DI(o, c, c') : $B'_- = \{o \in c\}$ et

$B'_+ = \{o \in c'\}$

$\cup \{o.a=v$ tel que $\{v\}$=Rest($o,a,\emptyset$) et $\{v\} \neq$Rest(o,a,[$c\ c'$[)$\}$

Pour supprimer le rattachement de l'instance à une de ses classes d'appartenance, la solution est un tout petit peu différente; en particulier, l'instance n'est attachée à aucune classe car elle y est attachée par ailleurs et les assertions ($o.a$=v) sont ajoutées en se basant sur l'intervalle entre la classe supprimée et la racine de la hiérarchie ou entre la classe supprimée et la première super-classe commune à sa classe d'attachement dans la hiérarchie et celle qui ne l'est plus. :

SA(o, c, c') :

Soit c''= PSSCC(c,c') si une telle classe existe

la racine de la hiérarchie de c sinon

$B'_- = \{o \in c\}$ et

$B'_+ = \{o.a=v$ tel que $\{v\}$=Rest($o,a,\emptyset$) et $\{v\} \neq$Rest(o,a,[$c\ c''$[)$\}$

c) Déplacer une classe sous une super-classe (DC)

Le lien de spécialisation entre les deux classes est coupé, la sous-classe étant attachée à l'une des super-classes.

Déplacer une classe conduit à la perte de certaines restrictions d'attributs dans la clôture déductive de la base révisée ; dans l'exemple de la Figure 39, le déplacement de la classe *c* entraîne la perte de l'assertion ($c.b \subseteq d$) car la base révisée est obtenue à partir de la forme première et donc cette assertion n'apparaît ni dans la forme première de la base révisée, ni dans sa clôture déductive car ce n'est plus une restriction posée sur une super-classe. lors du déplacement de classes, il faut donc préserver en particulier toutes les restrictions d'attributs qui ne doivent pas être supprimées pour lever l'inconsistance. Dans certaines stratégies de révision, certains restrictions d'attribut ne doivent pas être conservées car ce sont elles qui sont à l'origine du problème ; ces attributs sont rassemblés dans l'ensemble *Att*. Pour tous les autres attributs, la restriction dans la forme normale qu'ils avaient avant déplacement est à conserver dans la classe déplacée (Figure 39).

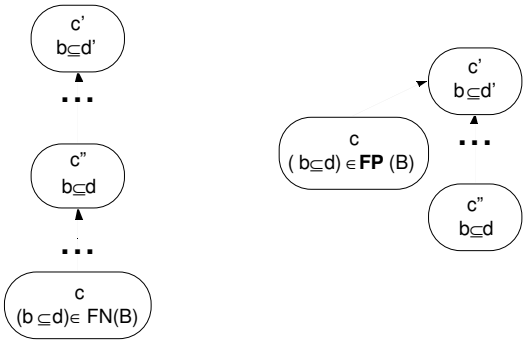


Figure 39 : Le déplacement de la classe *c* entraîne la perte de l'assertion indiquant la restriction de l'attribut b (qui appartient à la clôture déductive par la règle de (d-héritage)), c'est pourquoi cette assertion est ajoutée dans B'_+.

De plus, les assertions donnant la valeur de l'attribut d'une instance obtenues par la règle de (o-valuation) en nécessitant les restrictions d'attribut qui ont été supprimées (celles des classes comprises strictement entre *c* et *c'*) doivent également être conservées.

Formellement, le déplacement de classe s'exprime donc par :

pré-condition : $(c \leq^* c') \in \mathrm{FP}(B)$

DC(c, c', Att):

$B'_- = \{c \leq \sup(c)\}$

$B'_+ = \{c \leq c'\}$

$\cup \{(c.b \subseteq d)$, si $(c.b \subseteq d) \in \mathrm{FN}(B)$, $(c.b \subseteq d) \notin \mathrm{FP}(B)$, $(c'.b \subseteq d') \in \mathrm{FN}(B)$ et $d <: d'\}\}_{b \notin Att}$

$\cup \{o.a = v$ tel que $(o \in c) \in \mathrm{Cn}(B)$, $\{v\} = \mathrm{Rest}(o,a,\emptyset)$ et $\{v\} \neq \mathrm{Rest}(o,a,]c\ c'[\}_{a \in Att}$

d) Remonter des instances (RI)

Lorsqu'une classe est déplacée, à cause de l'hypothèse de mono-instanciation dans un point de vue, il existe deux possibilités pour les instances des classes déplacées :

- soit elles suivent la classe déplacée et le rattachement aux classes comprises entre la super-classe initiale et la super-classe finale est perdu,
- soit elles sont rattachées à la super-classe de la classe déplacée mais à ce moment-là le rattachement de l'instance aux sous-classes de la classe déplacée sont perdues.

La notion de minimalité ne permet pas de choisir entre ces deux alternatives car elles conservent chacune des assertions que l'autre supprime. Les bases révisées correspondant aux deux seront donc proposées au concepteur.

La première alternative ne nécessite aucune modification de la forme première de la base, alors que la seconde nécessite les modifications décrites ci-dessous:

$RI(c) = \{DI(o,c',sup(c))$ si $\exists c' (c' \leq^* c) \in FP(B)\ (o \in c') \in FP(B)\}$

Si certaines instances ne doivent pas être remontées, elles seront spécifiées de la manière suivante : $RI(c) \backslash \{o_i\}$ et les instances de cet ensemble ne devront donc pas être touchées.

e) Agrandir une restriction d'attribut (ARA)

Pour agrandir le domaine d'un attribut dans une classe, il s'agit non seulement de modifier les assertions concernant le domaine de l'attribut dans la classe donnée (en supprimant la restriction précédente et en la remplaçant par l'union avec le nouveau domaine) mais également d'élargir les domaines d'attributs de toutes les super-classes.

Dans la suite, c_m=PCP(c,a,d)

Quelles sont les assertions à préserver ?

- Des assertions étaient déductibles de la règle de (o-valuation) et ne le sont plus à cause du retrait de l'une des assertions de restriction d'attribut : si les nouvelles restrictions posées sur les classes entre c et c_{m-1} ne permettent pas de déduire de valeur pour l'attribut de l'instance.
- Des restrictions d'attributs sur les sous-classes d'une classe dont la restriction est modifiée peuvent être perdues alors qu'elles sont acceptables dans la base révisée. Les classes touchées sont celles qui sont sous-classes d'une classe modifiée sans être en lien de spécialisation avec c et qui héritait d'une restriction supprimée.

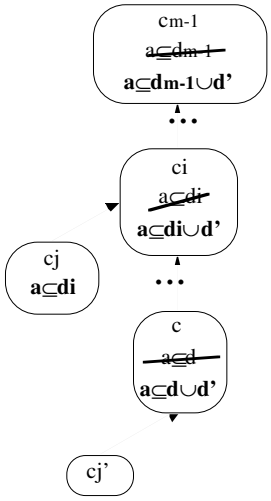


Figure 40 : la classe *cj* héritait de la restriction posée sur *ci*; après la suppression de celle-ci, afin d'avoir une base révisée minimale, la restriction est posée sur la classe *cj*. La classe *cj'* étant une sous-classe de *c* ne conserve pas sa restriction précédente, car la présente solution élémentaire peut avoir pour objectif de supprimer cette assertion; dans le cas contraire, les assertions correspondantes sont ajoutées dans la solution globale.

Formellement, l'agrandissement de restriction d'attribut est :

ARA(c, a, d)

$B'_- = \{(c_i.a \subseteq d_i)$, t.q. $\quad c_i \in [c\ c_m[$ et $(c_i.a \subseteq d_i) \in \mathrm{FP}(B)\}$ et

$B'_+ = \{(c_i.a \subseteq d_i \cup d)$, t.q. $c_i \in [c\ c_m[, (c_i.a \subseteq d_i) \in \mathrm{FP}(B)$ et $(c_m.a \subseteq d_i \cup d) \notin \mathrm{Cn}(B)\}$

$\cup \{o.a = v$ tel que $\{v\} = \mathrm{Rest}(o,a,\emptyset)$

et $\{v\} \neq \mathrm{Rest}(o,a,[c\ c_m[) \wedge (d \vee \mathrm{Rest}(o,a,\{c''$ t.q. $c'' \notin [c\ c_m[\}))$

$\cup \{(c_j.a \subseteq d_i)$, t.q. $(c_j \leq c_i) \in \mathrm{FP}(B)$, $c_i \in]c\ c_m[$, $\neg(c \leq^* c_j) \in \mathrm{FP}(B)$, $(c_i.a \subseteq d_i) \in \mathrm{FN}(B)$ et $(c_j.a \subseteq d_i) \in \mathrm{FN}(B)\}$

Une remarque s'impose : pour obtenir une base acceptable, il suffit que l'intersection des domaines d'une classe et de ses super-classes soit non nulle; lorsqu'une incohérence est produite par l'ajout d'un nouveau domaine d'un attribut d'une classe qui a une intersection vide avec celui dans la forme normale, l'une des possibilités est d'élargir ce domaine ainsi que le domaine de cet attribut dans les super-classes. L'élargissement minimal correspond à l'ajout d'une valeur, car cela suffit pour que l'intersection soit non vide. Cependant, le choix de cette valeur serait tout à fait arbitraire, c'est pourquoi nous avons décidé que les domaines élargis correspondraient à l'union de l'ancien domaine des classes et du nouveau domaine de l'attribut de la classe modifiée (Figure 41).

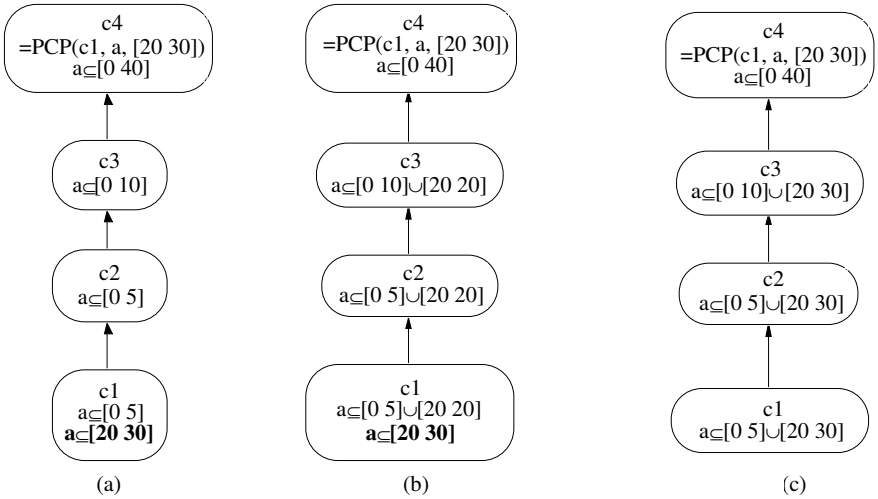


Figure 41 : (*a*) La classe *c1* est incohérente suite à l'ajout de ($c_1.a \subseteq$[20 30]); (*b*) pour lever cette incohérence, l'ajout de l'intervalle [20 20] dans les restrictions d'attributs des super-classes problématiques suffit pour obtenir une base révisée consistante; cependant, pourquoi choisir 20, et pas 21 ou 30 ? (*c*) C'est pourquoi, on adopte une stratégie qui consiste à ajouter tout le nouveau domaine dans les restrictions de l'attribut dans les super-classes.

2. Solution globale

Afin de décrire la révision globale de la base suite à l'ajout d'une assertion conduisant à une base inconsistante ou incohérente, on utilise l'addition de solutions élémentaires (de l'un des types présentés ci-dessus) qui correspond à l'union des ensembles B'_- et B'_+ de chaque solution élémentaire:

Soit $S_1,\ldots S_n$ des solutions, $S=S_1+\ldots+S_n$ est la solution telle que :

$B'_- = \bigcup_i B'_{-i}$, B'_{-i} étant l'ensemble des suppressions correspondant à la solution S_i.

$B'_+ = \bigcup_i B'_{+i}$, B'_{+i} étant l'ensemble des ajouts correspondant à la solution S_i.

Toutes ces modifications (les ajouts en particulier) ont l'agréable propriété de ne pas soulever ni incohérence, ni inconsistance supplémentaire, c'est-à-dire que si la modification permet de résoudre le problème initial, aucun autre n'aura pu apparaître.

Pour chaque assertion, nous verrons quelles sont les inconsistances et incohérences qui peuvent être soulevées parmi celles qui sont possibles :

inconsistances :

(1) il existe o, a, v et v' tels que $(o.a=v) \in B$ et $(o.a=v') \in B$ et $v \neq v'$

(2) il existe o, c, a, d et v tels que $B \vdash o \in c$, $B \vdash c.a \subseteq d$, $(o.a=v) \in \mathrm{FP}(B)$ et $\neg v{:}d$

(3) il existe o, $\{c_i\}$, a tels que $B \vdash o \in c_i$, $B \vdash c_i.a \subseteq d_i$ et $\bigwedge_i d_i = \bot$

(4) il existe o, c, c', c'' tels que $(o \in c) \in \mathrm{FP}(B)$, $(o \in c') \in \mathrm{FP}(B)$, $(c \leq^* c'') \in \mathrm{FP}(B)$ et $(c' \leq^* c'') \in \mathrm{FP}(B)$

ou incohérences :

(5) il existe c, a tels que $B \vdash c.a \subseteq \bot$

Remarquons que l'inconsistance de type (4) n'apparaît pas si une instance est attachée à deux classes qui sont liées par relation de spécialisation, car dans ce cas, il ne reste qu'une assertion de rattachement de l'instance dans la forme première.

3. L'assertion ($o.a=v$)

a) Problèmes soulevés

Cette assertion intervient directement dans des inconsistances du type (1) et (2) et ne permet de déduire aucune nouvelle assertion, car ce type d'assertion n'apparaît pas dans prémisses des règles d'inférence. Donc les inconsistances suivantes sont les seules possibles :

(1) il existe v' tels que $(o.a=v') \in \mathrm{FP}(B)'$ et $v \neq v'$

(2) il existe c_i et d_i tels que $B \vdash o \in c_i$, $B \vdash c_i.a \subseteq d_i$ et $\neg v : d_i$

b) Solutions élémentaires

Pour résoudre (1)

(solution 1) – *supprimer la valeur de l'attribut*

Si $(o.a=v') \in \mathrm{FP}(B)$, alors $\mathrm{S1}=\mathrm{S}(o,a,v')$

Pour résoudre (2)

Soit $(o \in c) \in \mathrm{FP}(B)$, on note alors $c_n=\mathrm{PCI}(c,a,\{v\})$ et $c_m=\mathrm{PCP}(c,a,\{v\})$

(solution 2) – élargir les domaines d'attribut

On peut élargir, en y ajoutant la valeur v, tous les domaines des attributs des super-classes appartenant à la branche entre la classe problématique la plus spécifique et celle qui ne pose pas de problème.

$\mathrm{S2}=\mathrm{ARA}(c_n, a, \{v\})$

(solution 3) – *déplacer une super-classe de la classe d'appartenance qui ne contient pas de restriction d'attribut*

Dans le cas où la classe d'appartenance de l'instance n'a pas de restriction pour l'attribut, c et c_n sont deux classes distinctes et on peut donc couper la chaîne entre ces deux classes en déplaçant l'une des classes sous la classe possible la plus spécifique.

Il s'agit de conserver les domaines les plus restreints pour l'attribut a, aussi bien dans la classe déplacée (le plus restreint étant alors son domaine dans la clôture déductive, c'est-à-dire le premier domaine défini dans les super-classes auquel est ajouté la valeur d'attribut de l'instance), que dans ses sous-classes (qui conservent les restrictions de la super-classe de c_i si elles n'en faisaient pas elles-mêmes).

Suite au déplacement de classes, on propose éventuellement de déplacer les instances appartenant à la classe déplacée, sauf pour o car si on applique le déplacement d'instance sur o, l'inconsistance subsiste (Figure 42).

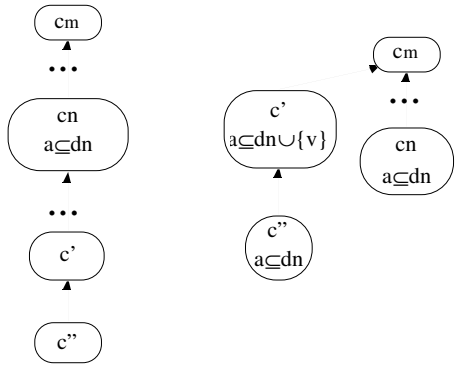


Figure 42 : Illustration du déplacement de classe et des modifications faites pour l'attribut a de la classe déplacée et de ses sous-classes.

Soit $c' \in]c \text{ sous}(c_n, c)]$:

S3= DC(c', c_m, {a})

\+ $B'_- = \emptyset$

$B'_+ = \{c'.a \subseteq d_n \cup \{v\}$ si $(c_n.a \subseteq d_n) \in \text{FP}(B)$ et $(c_m.a \subseteq d_n \cup \{v\}) \notin \text{Cn}(B)\}$

$\cup \{c''.a \subseteq d_n$ si $(c'' \leq c') \in \text{FP}(B)$ et il n'existe pas d'' t.q. $(c''.a \subseteq d'') \in \text{FP}(B)\}$

\+ éventuellement RI(c_i)\{o}

(solution 4) – *déplacer une super-classe de la classe d'appartenance et modifier les restrictions d'attribut dans les classes intermédiaires*

La chaîne entre c et sous(c_m,c)=c_{m-1} peut être coupée en déplaçant une classe entre c_n et c_m sous la super-classe possible la plus spécifique et en modifiant les restrictions d'attribut des classes déplacées.

La solution se décompose en :

- déplacement de la classe et déplacement éventuel des instances de la classe déplacée ;
- agrandissement des restrictions d'attribut ;
- préservation du maximum de restrictions sur l'attribut a dans la classe déplacée (en augmentant simplement le domaine en y ajoutant la valeur v) ou dans ses sous-classes (en conservant les restrictions posées).

Soit $c_i \in]c\ c_{m-1}[$ (déplacer la classe c_{m-1} revient à ne rien faire)

S4=DC(c_i, c_m, a) + éventuellement RI(c_i) + ARA(c_n, c_i, a, $\{v\}$)

La différence avec S3 est que la chaîne est coupée au-dessus de la classe inconsistante la plus spécifique, et donc que des restrictions d'attribut doivent être agrandies "en dessous" de la coupure (entre les classes c_n et c_i) (Figure 43).

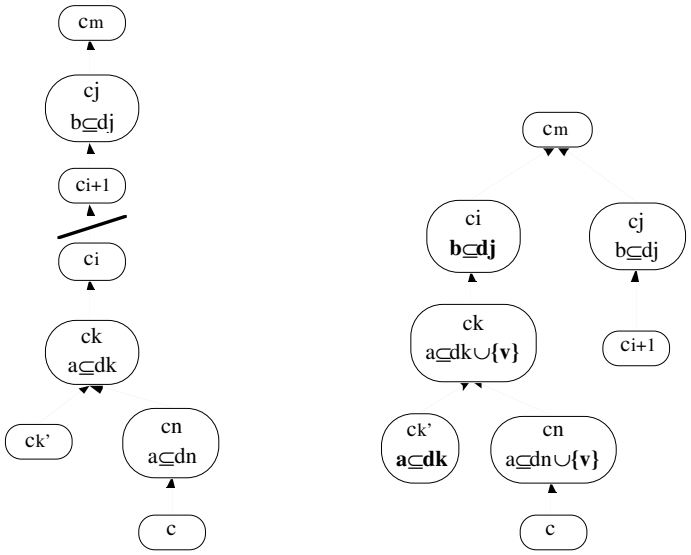


Figure 43 : Exemple de révision par déplacement d'une classe et illustration des différentes modifications effectuées, en particulier sur les restrictions d'attribut.

(solution 5) – *déplacer l'instance et modifier les restrictions d'attribut dans les super-classes*

L'instance peut être déplacée dans une classe et les restrictions des attributs modifiées dans les super-classes. Remarquons que si l'instance est déplacée sous c_m, aucune restriction d'attribut n'est modifiée (le second terme est vide).

Soit c_p telle que $(c_{p-1}.a \subseteq d_{p-1}) \in \mathrm{FP}(B)$ et $c_p \in]c_n\ c_m]$

S5= DI(o, c, c_p) + ARA(c_p, a, $\{v\}$)

La condition $(c_{p-1}.a \subseteq d_{p-1}) \in \mathrm{FP}(B)$ est nécessaire pour que la base révisée soit minimale; en effet, si cette condition n'est pas satisfaite, l'instance peut être attachée sous la sous-classe c_{p-1} en ne nécessitant pas plus de modification de restrictions d'attributs.

c) Solution globale

La solution est composée par ajout des différentes solutions élémentaires.

1. Si $(o.a=v') \in \mathrm{FP}(B)$, alors S=S1, sinon S=$\{ B'_- = \emptyset; B'_+ = \emptyset \}$
2. Si des inconsistances du type (2) existent, pour chaque classe d'appartenance de o qui est impliquée dans une inconsistance : S=S+(S2 ou S3 ou S4 ou S5)

La principale limitation à cette solution globale est qu'il manque une combinaison des trois solutions : déplacement de l'instance, d'une classe et modification des restrictions d'attribut. L'utilisateur s'y retrouverait-il ?

4. L'assertion ($o \in c$)

a) Problèmes soulevés

Cette assertion peut soulever directement des inconsistances du type (2), (3) et (4). De plus, on peut déduire grâce à elle des assertions du type ($o.a=v$), mais sans soulever d'inconsistance d'un autre type (en particulier dans (1), de telles assertions doivent appartenir à la base et non être déduites).

Il ne peut exister une inconsistance de type (4) qu'avec une seule classe car la base est acceptable avant l'ajout de ($o \in c$).

(2) $\{o.a_i=v_i, (c.a_i \subseteq d_i) \in \mathrm{FN}(B)$ et $\neg v_i : d_i\}_{i \in I}$
(3) $\{\{c_i\}, a$ tels que $B \vdash o \in c_i, B \vdash c_i.a \subseteq d_i$ et $\bigwedge_i d_i = \bot\}$

(4) il existe c', c'' tels que $(o \in c') \in \mathrm{FP}(B)$, $(c \leq^* c'') \in \mathrm{FP}(B)$ et $(c' \leq^* c'') \in \mathrm{FP}(B)$

b) Solutions élémentaires

Ci-dessous, nous donnons les solutions possibles pour résoudre chaque type d'inconsistance ; les numéros correspondent au numéro de la solution qui est détaillée dans la suite.

Les inconsistances de type (2) peuvent être résolues par les stratégies suivantes :

(1) couper la chaîne entre c et la classe dont la restriction d'attribut provoque l'inconsistance.

(2) élargir les restrictions d'attribut,

(3) supprimer la valeur des attributs de l'instance,

Il est bien entendu possible de combiner ces trois types de stratégies pour les différents attributs. Pour un attribut donné, nous ne prenons pas en compte la modification des restrictions d'attributs et la coupe de la chaîne.

Les inconsistances de type (3) peuvent être résolues par les stratégies suivantes :

(1) élargir les restrictions d'attribut de certaines classes d'appartenance,

(2) supprimer la valeur des attributs de l'instance,

(3) supprimer l'appartenance à certaines classes.

Les deux premières solutions sont indépendantes pour chaque attribut, mais la troisième peut résoudre l'inconsistance pour plusieurs attributs.

L'inconsistance de type (4) est résolue simplement en supprimant l'appartenance à la classe qui est dans le même point de vue.

Le Tableau 2 résume les solutions possibles pour chaque type d'inconsistance.

		Inconsistance		
		(2)	(3)	(4)
Solution	(1)	OK		
	(2)	OK	OK	
	(3)	OK	OK	
	(4)		OK	OK

Tableau 2 : résumé des types de solutions qui résolvent un type d'inconsistance donné.

La résolution de ces trois types d'inconsistance est indépendante car elles concernent des hiérarchies différentes. Il faut cependant résoudre l'inconsistance (4) en premier pour ne plus avoir à tenir compte des restrictions d'attributs posées par la classe qui ne sera plus classe d'appartenance de l'instance.

Soient c_{mi}=PCP(c,a_i,{v_i}) la classe la plus spécifique qui est compatible avec la valeur v_i et c_{ni}=PCI(c,a_i,{v_i}) la classe la plus spécifique qui n'est pas compatible avec la valeur v_i.

La résolution est indépendante pour chaque attribut, sauf si les liens de spécialisation entre classes sont modifiées. Il faut alors effectuer le déplacement en premier et traiter ensuite les attributs qui seront encore inconsistants. En effet, la résolution pour un attribut par déplacement de classe peut résoudre l'inconsistance pour d'autres attributs.

(solution 1) – *déplacer une classe sous PCP*

On choisit de déplacer une classe, notée c_n (qui est une PCI pour l'un des attributs) sous la classe PCP relative à un autre attribut (éventuellement le même) c_{mi}, notée c_m. On aurait pu choisir le déplacement d'une classe quelconque sous une classe quelconque, mais cette solution n'aurait pas été minimale, car il aurait toujours été possible de définir une solution faisant intervenir l'une des classes PCP ou PCI et qui aurait été plus proche, comme cela est exprimé par la Proposition 23.

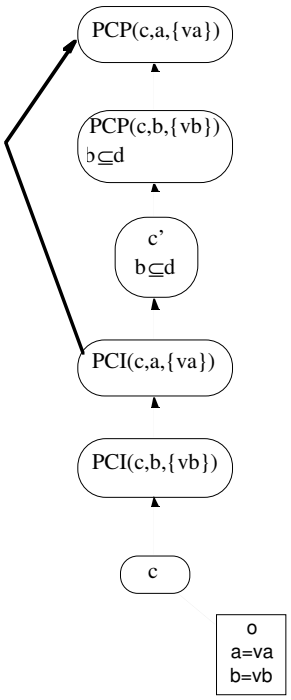


Figure 44 : Lors du déplacement de classe, la restriction de l'attribut *b* posée dans la classe *c'* ne doit pas être conservée dans la classe déplacée sous peine de ne pas résoudre l'inconsistance pour l'attribut *b*.

L'ensemble d'attributs pour lesquelles l'inconsistance est levée est alors :

$I1=\{j\in I$ t.q. $B\vdash c_n\leq \mathrm{PCI}(c,o,a_j,v)$ et $B\vdash \mathrm{PCP}(c,o,a_j,v)\leq c_m\}$, c'est-à-dire tous les attributs dont les classes aux restrictions problématiques (celles comprises entre PCI et PCP) ne sont plus super-classes de *c*. Pour ces attributs, la restriction d'attribut ne doit pas être conservée, sinon l'inconsistance n'est pas résolue, c'est pour cela que l'ensemble *I1* est passé en argument de DC (Figure 44).

S1(I1)=DC(c_n, c_m, I1) + éventuellement RI(c_n)

(solution 2) – *modifier les restrictions d'attribut*

Pour les attributs restants, il est possible de modifier les restrictions d'attributs des classes qui étaient initialement comprises entre c_{ni} et c_{mi}.

S2(a_i)=ARA(c_{ni}, a_i, {v_i})

(solution 3) – *supprimer la valeur d'attribut de l'instance*
Il est aussi possible de supprimer la valeur de l'attribut de l'instance.

S3(a_i)=S(o,a_i,v_i)

(solution 4) – *supprimer l'appartenance de l'instance à un ensemble de classes*
S4({c_i})={SA(o,c_i,racine(c))}

c) Solution globale

1. Pour l'inconsistance de type (4) :

SA(o, c, c')

2. Pour résoudre les inconsistances de type (3) :

a) Choisir l'ensemble C-suppr des classes qui ne seront plus classes d'appartenance : pour $c\in$ C-suppr, DI(o, c, *racine(c)*)}

b) Pour chaque attribut a_i qui apparaît encore dans une inconsistance de type (3) :
S3(a_i)
ou donner à l'utilisateur les différents domaines des classes, et il fournit le nouveau domaine pour cet attribut dans toutes les classes d'appartenance

3. Pour les inconsistances de type (2), la solution consiste à partitionner l'ensemble des attributs inconsistants en trois sous-ensembles I1, I2 et I3, sachant que l'ensemble I1 ne peut être quelconque; si $c_{n1}\leq c_{n2}$ et $c_{m2}\leq c_{m1}$, il ne sera pas possible d'avoir dans cet ensemble l'attribut 1 sans l'attribut 2.

a) Choisir la classe c_n dans l'ensemble {PCP(c,a_i,{v_i}), $i\in I$} et choisir la classe c_m dans l'ensemble {PCI(c,a_i,{v_i}), $i\in I$}. L'ensemble I1 est alors défini par $I1$={$j\in I$ t.q. $B\vdash c_n\leq$PCI(c,o,a_j,v) et $B\vdash$PCP(c,o,a_j,v)$\leq c_m$}.
b) Partitionner l'ensemble I–I1 en deux sous-ensembles I2 et I3.
c) Déplacer c_n sous c_m : S1(I1).
d) Appliquer S2 sur I2 et S3 sur I3.

Quelques solutions particulières du point 3 de l'algorithme :

- pas de déplacement (c_n=c_m) : résolution uniquement locale pour chaque attribut en supprimant la valeur de l'attribut ou en agrandissant les domaines

- ou au contraire résolution des problèmes uniquement par déplacement de classes: déplacement de la classe la plus spécifique inconsistante PCI sous la plus haute classe possible PCP pour l'ensemble des attributs.

5. L'assertion ($c \leq c'$)

Dans TROPES, la migration de classe n'est pas autorisée; les assertions du type ($c \leq c'$) sont donc uniquement utilisée pour indiquer la super-classe d'une classe lors de son introduction dans la base, donc lorsque la classe ne possède ni sous-classe, ni instance, ni restriction d'attributs. Il n'y aura donc pas de problème à résoudre suite à ce genre d'ajout. Un cas toléré est le déplacement sous une super-classe qui ne peut soulever ni inconsistance, ni incohérence, car les restrictions posées sur les attributs sont moins strictes. Sous cette hypothèse, aucun problème ne peut apparaître suite à l'ajout de cette assertion.

Sans cette hypothèse, les problèmes qui peuvent apparaître sont des incohérences dans les sous-classes ainsi que des instances inconsistantes. Pour résoudre ces problèmes, voici une ébauche de solution :

1. Localiser les premières sous-classes qui sont incohérentes dans chaque branche.
2. Résoudre ces incohérences en déplaçant les classes ou en modifiant les restrictions d'attribut.
3. Si des incohérences persistent dans les sous-classes, résoudre récursivement.
4. Résoudre alors les inconsistances qui persistent pour les instances car certaines inconsistances auront pu être résolues par la résolution des incohérences.

Il faut résoudre d'abord les incohérences puis les inconsistances car la résolution des premières peut résoudre les secondes, alors que le contraire n'est pas vrai.

6. L'assertion ($c.a \subseteq d$)

a) Problèmes soulevés

Cette assertion apparaît explicitement dans les types d'inconsistances (2) et (3). De plus, elle permet éventuellement de déduire une assertion du type ($c.a \subseteq \bot$) portant sur la classe elle-même ou l'une de ses sous-classes. Les problèmes pouvant apparaître sont donc des classes incohérentes et des inconsistances dans tout le sous-arbre de la classe modifiée. L'ensemble des classes incohérentes est :

$C = \{c_i \text{ t.q. } (c_i.a \subseteq d_i) \in \mathrm{FP}(B), (c_i \leq^* c) \in \mathrm{FP}(B) \text{ et } d_i \wedge d = \bot\}$
$\cup \{c \text{ si } \exists c_i\ (c_i.a \subseteq d_i) \in \mathrm{FP}(B), (c \leq^* c_i) \in \mathrm{FP}(B) \text{ et } d_i \wedge d = \bot\}$

L'ensemble des inconsistances est alors :

$I=I1 \cup I2$ avec $I1=\{o, B \vdash o \in c_i, c_i \in C\}$ et
$I2=\{o, B \vdash o \in c, (o.a=v) \in \mathrm{FP}(B), \neg v{:}d.\}$.

b) Solutions élémentaires

Les solutions élémentaires pour les différents problèmes sont données, le numéro faisant référence aux solutions détaillées par la suite.

Si la classe *c* est incohérente :

1. Modifier le domaine dans les super-classes de la classe *c* et résoudre les inconsistances et incohérences dans le sous-arbre ;
2. Déplacer une classe comprise entre *c* et la première classe qui pose problème. Les sous-classes sont déplacées avec et leur domaine est modifié ;
3. Identique à la solution précédente, sauf que les sous-classes ne sont pas déplacées.

Pour les autres classes incohérentes :

4. Déplacer la classe avec ses sous-classes et modifier le domaine des sous-classes ;
5. Déplacer la classe sans ses sous-classes; elles sont donc déplacées sous l'ancienne super-classe de leur super-classe (le problème est alors résolu pour les sous-classes, vu que la base était acceptable avant cette dernière assertion ;
6. Supprimer la restriction d'attribut.

On ne peut pas envisager de modifier les domaines des super-classes, car les instances sont dans une sous-classe de *c* et donc les modifications de restriction d'attribut devraient modifier également la restriction d'attribut de *c*, ce qui n'est pas possible, car l'assertion ajoutée doit appartenir à la base révisée. Il faut donc obligatoirement les déplacer si on veut préserver leur valeur d'attribut. Il reste alors à résoudre les inconsistances sur les instances. Si sa classe d'appartenance a été déplacée, il n'y a plus d'inconsistance après la résolution des incohérences, sinon, il est possible de :

7. de supprimer leur valeur.
8. les déplacer ;

Formellement :

Soit $c' = \sup(c)$

Comme la description d'une solution générale serait un peu trop lourde, nous allons détailler des solutions pour un sous-ensemble d'incohérences et d'inconsistances et donnerons alors un algorithme permettant de trouver une base révisée.

Si $c \in C$

Soient c_m=PCP(c,a,d) et c_n=PCI(c,a,d)

(solution 1) – *élargir les restrictions d'attribut dans les super-classes*

Élargir les restrictions d'attribut de toutes les super-classes qui le nécessitent.

S1=ARA(c_n,a,d)

(solution 2) – *déplacer la classe*

Déplacer la classe et modifier les restrictions d'attribut des super-classes de la nouvelle super-classe de c et préserver les restrictions d'attribut qui ne sont pas concernées (dans les sous-classes de la classe déplacée dans les branches ne contenant pas c; d_n étant la restriction d'attribut la plus proche) (Figure 45).

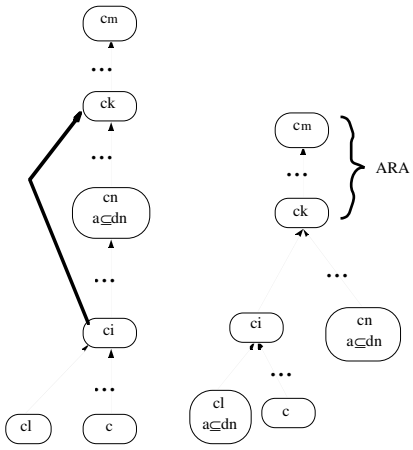


Figure 45 : Illustration du déplacement de classe proposée par la solution 2, avec préservation des restrictions d'attribut.

Soient $c_i \in [c\ c_n[$ et $c_k \in [\sup(c_n) c_m]$ avec (sous(c_k,c_n).$a \subseteq d_{k-1}$) $\in$ FP(B).

S2=DC(c_i,c_k,{a}) + ARA(c_k,a,d) + éventuellement RI()

\+ $B'_- = \emptyset$

$B'_+ = \{c_l.a \subseteq d_n$ si $(c_l \leq c_i) \in$ FP(B), $(c \leq c_l) \notin$ FP(B) et $(c_l.a \subseteq d_n) \in$ FN(B)$\}$

Un cas particulier de cette solution :

Si $c_k=c_m$, aucune restriction d'attribut n'est modifiée

S2'=DC(c_i,c_m,{a}) + éventuellement RI()

\+ $B'_- = \emptyset$

$B'_+ = \{c_l.a \subseteq d_n$ si $(c_l \leq c_i) \in$ FP(B), $(c \leq c_l) \notin$ FP(B) et $(c_l.a \subseteq d_n) \in$ FN(B)$\}$

(solution 3) – *déplacer la classe sans déplacer les sous-classes*

Identique à la solution précédente, mais les sous-classes ne sont pas déplacées avec la classe (ou autrement dit, elles sont déplacées sous l'ancienne super-classe de c).

S3 = S2 + {DC(c_i, sup(c), a), tel que ($c_i \leq c$)$\in$ FP(B)}

Pour $c_i \neq c \in C$, qui n'a pas de super-classe dans l'ensemble C.

(solution 4) – *déplacer la classe*

Déplacer la classe et préserver ainsi les restrictions d'attributs qui ne sont pas concernées. Cette solution est différente de (solution 2) car on déplace la classe elle-même et non l'une de ses super-classes.

S4=DC(c_i, c_m, a)

(solution 5) – *déplacer la classe sans ses sous-classes*

Elle est identique à la précédente sauf que les sous-classes ne sont pas déplacées avec la classe (c'est le même parallèle qu'entre les solutions 2 et 3).

S5=S4 + {DC(c_j,c",a), tel que ($c_j \leq c_i$)$\in$ FP(B)} avec ($c_i \leq c$")$\in$ FP(B)

(solution 6) – *supprimer la restriction d'attribut de classe*

Supprimer la restriction d'attribut

S6= $B'_- = \{c_i.a \subseteq d_i\}$

$B'_+ = \{o.a=v$ tel que ($o \in c_i$)$\in$ Cn(B), $\{v\}$=Rest(o,a,Ø) et $\{v\} \neq$ Rest(o,a,$\{c_i\}$)$\}$

Cette restriction supprimée n'est pas remise dans les sous-classes car cela consisterait à répercuter l'incohérence

Pour les instances$\in$ I2 :

(solution 7) – *supprimer la valeur d'attribut de l'instance*

S7=Suppr(o, a, v)

(solution 8) – *déplacer l'instance*

S8=DI(o, c_i, c_m), si ($o \in c_i$)$\in$ FP(B)

On ne peut pas envisager de modifier les domaines des super-classes, car les instances sont dans une sous-classe de c et donc les modifications de restriction d'attribut devraient modifier également la restriction d'attribut de c, ce qui n'est pas possible car l'assertion doit appartenir à la base révisée; il faut donc obligatoirement les déplacer si on veut préserver leur valeur d'attribut.

c) Solution globale

1. Si $c \in C$, S=(S1 ou S2 ou S2' ou S3) ($C \leftarrow C \setminus \{c\}$)
2. Pour $c_i \in C$, telle qu'elle n'a pas de super-classe dans C :
   S=S+S4 ($C \leftarrow C \setminus \{c'_i, B \vdash c'_i \leq c_i\}$, $I \leftarrow I \setminus \{o, (o \in c'_i) \in \mathrm{FP}(B), B \vdash c'_i \leq c_i\}$)
   ou S=S+S5 ($C \leftarrow C \setminus \{c_i\}$, $I \leftarrow I \setminus \{o,$ $(o \in c_i) \in \mathrm{FP}(B)$ et $\neg B \vdash o \in c_j$, $c_j \in C$, $(o.a=v) \notin \mathrm{FP}(B)\}$)
   ou S=S+S6 ($C \leftarrow C \setminus \{c_i\}$, $I \leftarrow I \setminus \{o,$ $(o \in c_i) \in \mathrm{FP}(B)$ et $\neg B \vdash o \in c_j$, $c_j \in C$, $(o.a=v) \notin \mathrm{FP}(B)\}$)
3. Répéter 2 tant que $C \neq \emptyset$. ($C=\emptyset$ et $I1=\emptyset$)
4. Pour $o \in I2$, S=S+S7 ou S=S+S8

C. Algorithme sur langage avec référence aux instances

Nous allons à présent examiner les solutions algorithmiques pour la révision dans des bases exprimées dans le langage autorisant les références aux instances. La principale difficulté est due aux règles d'inférence supplémentaires, car une nouvelle assertion permet de déduire des assertions beaucoup moins localisées dans la base. En particulier, inconsistances et incohérences peuvent se déclarer dans une toute autre hiérarchie que celle sur laquelle porte la dernière assertion. De plus, les modifications élémentaires de la base peuvent soulever d'autres problèmes dans la base. Alors que dans le cas simple présenté ci-dessus, des modifications telles que le déplacement d'une classe sous une super-classe ou le déplacement d'une instance sous une super-classe de son ancienne classe de rattachement ne pouvaient pas poser de problèmes supplémentaires, ce n'est plus le cas lorsque des références aux instances existent. Nous allons donc, dans un premier temps, préciser certaines définitions, avant de reprendre les modifications élémentaires et voir quels sont les problèmes qui peuvent survenir et la manière de les résoudre. L'étude de la complexité montre que l'algorithme permet toujours d'obtenir *une* base révisée en un temps polynomial (annexes C et D).

1. Remarques préliminaires

Rappelons que l'hypothèse de mono-spécialisation est posée et que l'on se place à présent dans le cadre du langage avec référence aux instances.

Nous verrons dans les parties 4, 5, 6, 7, 8 et 9 respectivement pour chaque assertion, quelles sont les inconsistances et incohérences qui peuvent être soulevées parmi celles qui sont possibles :

inconsistances :

(1) il existe $o \in O$, $a \in A$, $v \in V \cup O$ et $v' \in V \cup O$ tels que $(o.a=v) \in B$, $(o.a=v') \in B$ et $v \neq v'$

(2) il existe $o \in O$, $c \in C$, $a \in A$, $d \in T$ et $v \in V$ tels que $B \vdash o \in c$, $B \vdash c.a \subseteq d$, $(o.a=v) \in B$, et $\neg v:d$

(3) il existe $o \in O$, $\{c_i\} \subseteq C$, $\{a_i\} \subseteq A$ tels que $B \vdash o \in c_i$, $(c_i.a_1.a_2.\ldots a_n \subseteq^* d_{ij}) \in B$ et $\wedge_{ij} a_{ij} = \bot$

(4) ou il existe o, c, c', c'' tels que $(o \in c) \in \mathrm{Cn}(B)$, $(o \in c') \in \mathrm{Cn}(B)$, $(c \leq^* c'') \in \mathrm{FP}(B)$ et $(c' \leq^* c'') \in \mathrm{FP}(B)$

ou incohérences :

(5) il existe $c \in C$ et $a_1, a_2, \ldots, a_n \in A$ tels que $B \vdash (c.a_1.a_2.\ldots a_n \subseteq^* \bot)$.

Afin de mieux expliquer les stratégies de révision, nous commençons comme précédemment par énumérer les modifications possibles de la base, de la forme première de la base en l'occurrence. Vu l'hypothèse de mono-spécialisation, les hiérarchies de classes sont en fait des arbres. On appellera sous-arbre d'une classe, la partie de l'arbre composée de toutes les sous-classes de la classe.

2. Modifications élémentaires de la forme première

Nous aurons besoin de savoir si une assertion du type $(o \in c)$ est déductible par la règle de ($\in$-inférence). E-réf(o,c) est l'ensemble des instances qui référencent o et qui permette de déduire l'appartenance de l'instance o à la classe c par la règle de ($\in$-inférence).

E-réf(o,c)=$\{(o',a)$ t.q. $(o' \in c') \in \mathrm{Cn}(B)$, $(c'.a \subseteq^* c) \in \mathrm{FP}(B)$ et $(o'.a=o) \in \mathrm{FP}(B)\}$

a) Suppression d'une valeur d'un attribut d'une instance

Si $v \in V$ et $(o.a=v) \in \mathrm{FP}(B)$, S$(o,a,v)$: $B'_- = \{o.a=v\}$ et $B'_+ = \emptyset$

Si $o' \in O$, S(o,a,o',CNC) : grâce à l'assertion $(o.a=o')$, on pouvait peut-être déduire des assertions du type $(o' \in c')$ par la règle de ($\in$-inférence). Il est donc possible que cette solution élémentaire supprime des assertions du type $(o' \in c')$; cette suppression peut être à bon escient, c'est-à-dire que c'est l'objectif de l'application de cette solution élémentaire, au contraire, ces suppressions peuvent ne pas être voulues et dans ce cas, il faut les préserver dans la base révisée. Les classes d'appartenance de *o'* qu'il ne faut pas conservées sont regroupées dans l'ensemble CNC passé en argument.

Les assertions $(o' \in c')$ qu'il faut conserver explicitement dans la base révisée doivent vérifiées :

- elles sont déductibles grâce à $(o.a=o')$ et pas grâce à d'autres instances qui référencent *o'*, ce qui s'exprime encore par E-réf(o',c')=$\{(o,a)\}$,

- elles doivent appartenir à la forme première (donc l'instance ne doit pas être attachée à une classe plus spécifique ni dans la forme première, ni déductible par ($\in$-inférence),
- la classe c' n'est pas sous-classe d'une classe de CNC.

$B'_- = \{o.a = o'\}$ et

$B'_+ = \{o' \in c'$ si E-réf$(o',c') = \{(o,a)\}$,

il n'existe pas $c'' \neq c'$ telle que $(o' \in c'') \in FP(B)$

ou $(c'' \leq^* c') \in FP(B)$ et E-réf$(o',c'') \neq \emptyset$

et il n'existe pas $c'' \in$ CNC t.q. $(c' \leq^* c'') \in FP(B)\}$

b) Agrandir une restriction d'attribut

Il est parfois nécessaire d'élargir une restriction d'attribut aussi bien quand l'attribut est restreint par un domaine que par une classe. L'agrandissement de restriction d'attribut lorsque l'attribut est restreint par un domaine se fait de manière identique à celui dans le langage simple. Aucune inconsistance supplémentaire ne peut apparaître.

Élargir la restriction d'un attribut qui référençait une classe revient à restreindre l'attribut par une super-classe de celle qui le restreignait précédemment. Le raisonnement est similaire à celui pour les domaines à la différence près que l'on peut définir les nouveaux domaines en faisant des unions avec les valeurs nécessaires, alors qu'il faut utiliser les classes existantes. De plus, afin d'avoir effectivement un élargissement de la restriction – ce qui est obligatoire car la clôture déductive de la base contractée est un sous-ensemble de la clôture déductive de la base initiale – la nouvelle classe est une super-classe de l'ancienne restriction. Cette nouvelle classe doit également être une super-classe soit de la nouvelle classe qui est référencée, soit de la classe d'appartenance d'une instance (Figure 46). Nous donnons ci-dessous une définition (pour chacun de ces deux cas) de la classe la plus spécifique conforme aux nouvelles restrictions.

DÉFINITION 33 : PREMIÈRE CLASSE POSSIBLE (AVEC UNE CLASSE)

PCP(c,a,c') est la super-classe la plus spécifique de la classe c dont la restriction pour l'attribut a est une super-classe de c', c'est-à-dire :

PCP$(c,a,c') = c_m$ t.q. pour tout c_i telle que $(c_m \leq^* c_i) \in FP(B)$ et $(c_i.a \subseteq c'_i) \in FP(B)$ alors $(c' \leq c'_i) \in FP(B)$.

et si $(\text{sous}(c_m,c).a \subseteq c'_{m-1}) \in FP(B)$ alors $\neg(FP(B) \vdash c' \leq c'_{m-1})$.

DÉFINITION 34: PREMIÈRE CLASSE POSSIBLE (AVEC UNE INSTANCE)

PCP$(c,a,\{o\})$ est la super-classe la plus spécifique de la classe c dont la restriction pour l'attribut a est compatible avec la classe d'appartenance de l'instance o , c'est-à-dire :

PCP(c,a,{o})= c_m t.q. pour tout c_i telle que FP(B) $\vdash c_m \leq c_i$ et $(c_i.a \subseteq c'_i) \in$ FP(B) alors FP(B)$\vdash o \in c'_i$.

et si (sous(c_m,c).$a \subseteq c'_{m-1}$)$\in$ FP(B) alors $\neg$(FP(B) $\vdash o \in c'_{m-1}$).

Ce n'est donc *pas* la *possibilité* d'appartenance d'une instance à une classe mais le rattachement effectif qui est pris en compte pour trouver la classe compatible avec l'instance.

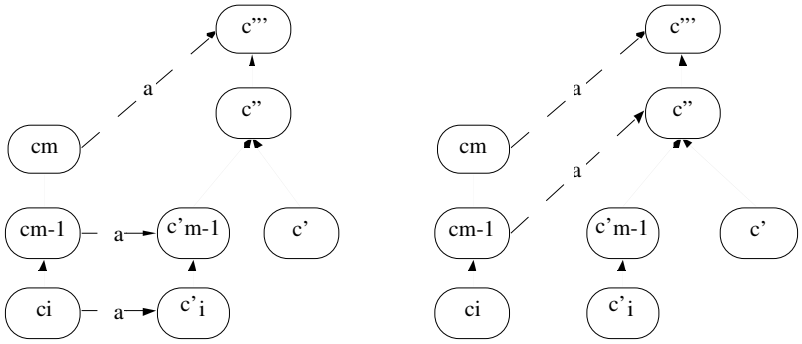


Figure 46 : Les classes c_i et c_{m-1} référençaient respectivement les classes c'_i et c'_{m-1}. Or, il s'agit d'étendre cette restriction afin qu'elles soient des super-classes de c', donc les attributs sont restreints par la super-classe commune la plus spécifique qui est c''. Cette classe est commune grâce à l'hypothèse de mono-spécialisation et il suffit donc de poser cette nouvelle restriction dans la plus haute super-classe posant problème, c'est-à-dire c_{m-1}.

Lors de l'agrandissement des restrictions d'attributs, des assertions du type ($o \in c$) peuvent être perdues; ce sont celles qui ne peuvent être déduites que grâce aux assertions ($c_i.a \subseteq c'_i$) qui sont supprimées. Dans B'_+, le rattachement à la classe la plus basse possible est conservé si celui-ci ne peut être déduit grâce à d'autres instances qui la référencent.

De plus, des restrictions d'attributs peuvent également être perdues dans les sous-classes des classes modifiées alors que ces restrictions sont à conserver si les classes n'appartiennent pas à la branche entre c et c_{m-1}. et si une nouvelle restriction n'est pas donnée dans la classe.

Avec c_m = PCP(c,a,c') et c_{m-1}=sous(c_m,c)

ARAC(c, a, c')

$B'_- = \{(c_i.a \subseteq c'_i), \forall i$ t.q. $c_i \in [c\ c_m[$ et $(c_i.a \subseteq c'_i) \in$ FP(B)$\}$ et

$B'_+ = \{(c_{m-1}.a \subseteq c''), c''$=PSSCC($c'$,$c'_{m-1}$) et $(c_m.a \subseteq c'') \notin$ Cn(B)$\}$

$\cup \{o' \in c'_i$ s'il existe un couple $(o,a) \in$ E-réf(o',c'_i) et si pour tous les tels couples (o,a) :

$(o \in c_i) \in$ Cn(B), $c_i \in [c\ c_m[$

et pour $(c_i'' \leq^* c_i') \in$ FP(B) et $c_i'' \neq c_i'$, E-réf(o',c''_i)=Ø et $(o' \in c''_i) \notin$ FP(B)$\}$

$\cup$ $\{(c_j.a \subseteq c'_i)$, t.q. $(c_j \leq c_i) \in$ FP(B), $c_i \in]c\ c_m[$, $\neg(c \leq^* c_j) \in$ FP(B), $(c_i.a \subseteq c'_i) \in$ FN(B) et $(c_j.a \subseteq c'_i) \in$ FN(B)$\}$

Cette modification de base ne soulève pas d'autre inconsistance ou incohérence.

ARAC(c, a, $\{o\}$) se définit de manière identique à la seule différence que c_m=PCP(c,a,$\{o\}$)

c) Résoudre les inconsistances soulevées par la présence de $o.a=o'$, $c.a \subseteq c'$ et ($o' \in c'$) inacceptable

Comme ce cas apparaît plusieurs fois, on traite dans ce paragraphe la solution globale permettant de le résoudre. L'hypothèse supplémentaire est que l'assertion ($o.a=o'$) ne peut être supprimée.

L'ensemble d'inconsistances est $I=I1 \cup I2$ avec :

$I1=\{(a_1, a_2,\ldots, a_n) \in A$ tels que $c'.a_1.a_2.\ldots.a_n \subseteq *d$, $o'.a_1.a_2.\ldots.a_n=*v$ et $\neg v:d\}$

$I2=\{(a_1, a_2,\ldots, a_n) \in A$ tels que $c'.a_1.a_2.\ldots.a_n \subseteq *c''$, $o'.a_1.a_2.\ldots.a_n=*o''$ $(o'' \in c''') \in$ FP(B), racine(c'')=racine(c''')$\}$

Les solutions permettant de résoudre indépendamment chacune de ces inconsistances sont :

(1) Couper la chaîne $o'.a_1.a_2.\ldots.a_n=*v$, en supprimant l'une des assertions $o_i.a_i=o_{i+1}$ avec $1 \leq i < n$ ou $o_n.a_n=v$:

S1(o_i,a_i,o_{i+1})=S(o_i,a_i,o_{i+1}) avec $1 \leq i < n$

S1(o_n,a_n,v)=S(o_n,a_n,v)

(2) Changer la restriction d'attribut de l'une des classes qui intervient dans $c'.a_1.a_2.\ldots.a_n \subseteq *d$:

soit $(a_1, a_2,\ldots, a_n) \in I1$:

S2(c_n,a_n,$\{v\}$)=ARA(c_n,a_n,$\{v\}$)

S2(c_i,a_i,c'_{i+1})=ARAC(c_i,a_i,c'_{i+1}) avec c'_{i+1} telle que :

$c'_{i+1}.a_{i+1}.\ldots.a_n \subseteq *d'$ et $v:d'$.

et pour toute c''_{i+1} t.q. $c''_{i+1} \in [c_{i+1}\ c'_{i+1}]$ et $c''_{i+1}.a_{i+1}.\ldots.a_n \subseteq *d''$, $\neg v:d''$

soit $(a_1, a_2,\ldots, a_n) \in I2$:

S2(c_i,a_i,c'_{i+1})=ARAC(c_i,a_i,c'_{i+1}) avec c'_{i+1} telle que :

pour toute c''' t.q. $c'_{i+1}.a_{i+1}.\ldots.a_n \subseteq *c'''$, $c''' \in$ [PSSCC(c'', c''') racine(c'')]

et pour toute c''_{i+1} t.q. $c''_{i+1} \in [c_{i+1}\ c'_{i+1}]$, il existe c''' t.q. $c''_{i+1}.a_{i+1}.\ldots.a_n \subseteq *c'''$, et $c''' \notin$ [PSSCC(c'', c''') racine(c'')]

(3) pour $I2$, il est également possible de supprimer l'appartenance de l'instance à la classe du même point de vue.

S3(o'') = SA(o'',c''',c'')

Mais la résolution pour les différentes inconsistantes n'est pas indépendante. En effet, la solution S1 résout le problème pour plusieurs valeurs d'attribut si les instances appartiennent à la même classe (Figure 47) et la solution S2 peut résoudre le problème pour toutes les chaînes d'attributs identiques à partir du rang *i* (Figure 48).

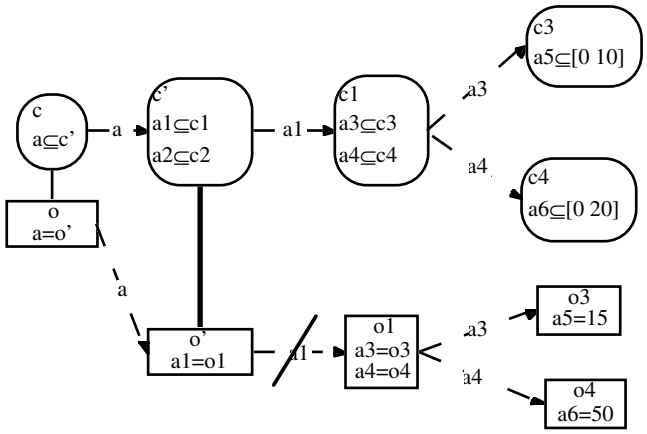


Figure 47 : Dans ce cas la solution S1 n'est pas indépendante pour résoudre les différents problèmes. En effet si on supprime o'.a1=o1, ceci résout les problèmes pour les chemins d'attribut (o.a.a1.a3.a5=15 avec c.a.a1.a3.a5⊆*[0 10]) et (o.a.a1.a4.a6=50 avec c.a.a1.a4.a6⊆*[0 20])

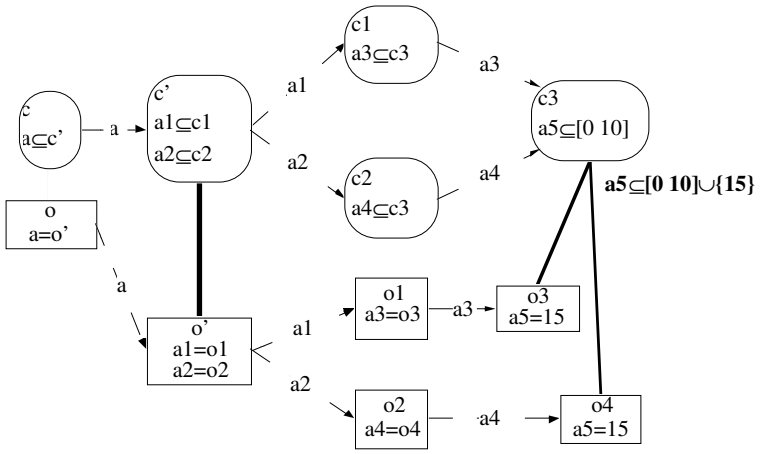


Figure 48 : Dans ce cas la solution S2 n'est pas indépendante pour chaque inconsistance. En effet, le fait de modifier le domaine de la classe c3 en [0 10]∪{15} résout le problème pour les chemins d'attributs (o.a.a1.a3.a5=15 avec c.a.a1.a3.a5⊆*[0 10]) et (o.a.a2.a4.a5=15 avec c.a.a2.a4.a5⊆*[0 10])

C'est pourquoi, nous envisageons une solution globale qui consiste tout d'abord à proposer la suppression des assertions du type (*o.a=o'*), puis à régler les problèmes restants par des modifications de restrictions d'attribut (Figure 49).

S-Référence1(*o*,*a*,*o'*,*c*) = S-réf1-o(*o'*) + S-réf1-c(*c*,*a*)

S-réf1-o(o') :

Pour chaque attribut a_i problématique de o'

S1(o',a_i,o_i) couper la chaîne en supprimant o'.a_i=o_i

ou S3(o_i) si o_i est impliquée dans I2

ou si o'.a_i=o'$_i$ avec o'i instance, S-réf1(o'$_i$)

ou pas de suppression du type (o.a=o') pour les instances référencées par o'$_i$

S-réf1-c(c,a) :

Pour toutes les classes c' telles que la restriction de $c.a$ par c' est encore problématique :

S2(c,a,c') et on rajoute les ($o.a$=o') (défaire (S1)) et les ($o' \in c'$) (défaire (S3)) qui ont été supprimées inutilement

ou si c' est une classe, pour chaque attribut a_i de c' S-réf1-c(c', a_i)

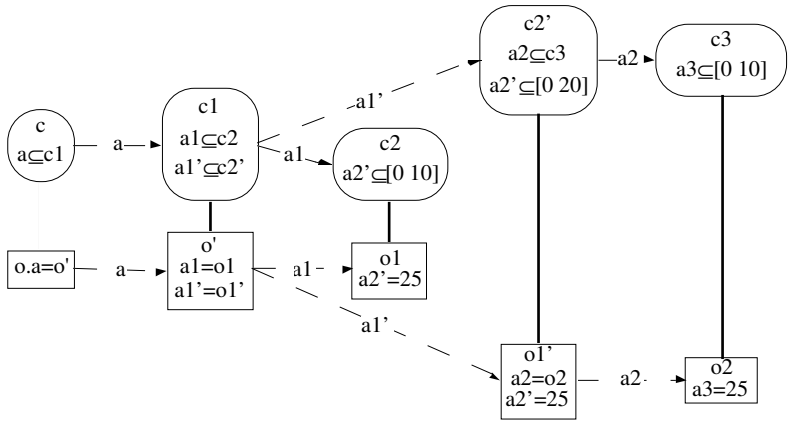


Figure 49 : I={o'.a1.a2'=25 et c.a.a1.a2'⊆[0 10] (par la classe c2); o'.a1'.a2'=25 et c.a.a1.a2'⊆[0 20] (par la classe c2'); o'.a1'.a2.a3=25 et c.a.a1'.a2.a3⊆[0 10] (par la classe c3)}. Les solutions sont alors :

(&) pour a1 : supprimer o'.a1=o1

ou pour a2' :supprimer o1.a2'=25

et pour a1' :supprimer o'.a1=o1' (**)

ou pour a2 : supprimer o1'.a2=o2

ou pour a3 :supprimer o2.a3=25 (*)

et pour a2' supprimer o1'.a2'=25

(&&) puis (si besoin) modifier la restriction de c.a

ou pour a1 : changer c1.a1 (*)

ou pour a2' : changer c2.a2' (**)

et pour a1' : changer c1.a1'

ou pour a2 : changer c2'.a2

ou pour a3 : changer c3.a3

et pour a2' : changer c2'.a2' (*)

En choisissant les solutions marquées de (*) (resp. (**)) dans la liste des possibilités associées à la Figure 49, la base obtenue est représentée dans la Figure 50 (resp. Figure 51).

Lors de la suppression d'une assertion du type ($c.a \subseteq c'$), il faut vérifier si des suppressions du type ($o.a=o'$) et ($o' \in c'$) ne deviennent pas superflues et, le cas échéant, remettre ces assertions dans la base[18]. Voyons un exemple sur la base illustrée par la Figure 48. Si dans la première étape de l'algorithme, on choisit de supprimer ($o_3.a_5$=15) et de ne pas toucher aux autres assertions de ce type. Il reste dans la seconde partie de l'algorithme à résoudre le problème pour ($o.a.a_2.a_4.a_5$=15, $c.a.a_2.a_4.a_5 \subseteq$*[0 10]), ce qui peut se résoudre en modifiant le domaine de c_3 en [0 10]$\cup${15}. L'assertion ($o_3.a_5$=15) est alors consistante dans la base et il est donc possible de la remettre.

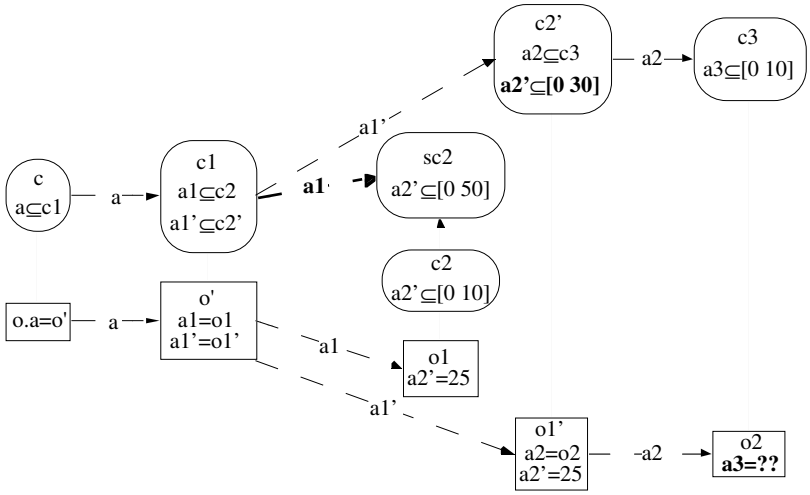
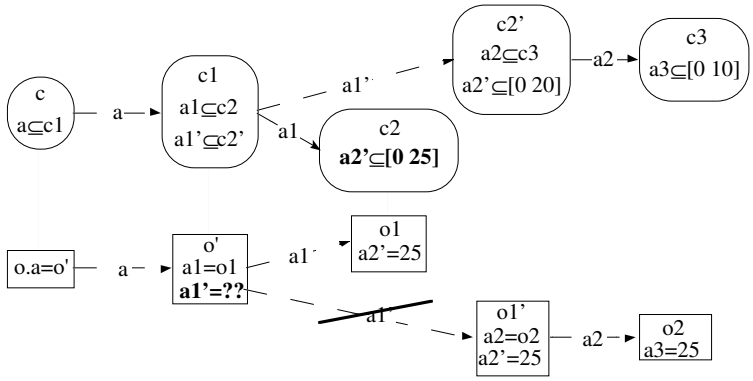


Figure 50 : Base obtenue en appliquant la révision indiquée par (*) dans la Figure 49.



[18] Toute idée d'algorithme ne nécessitant pas cette vérification est la bienvenue !

Figure 51 : Base obtenue en appliquant la révision indiquée par (**) dans la Figure 49.

d) Déplacer une instance dans une super-classe

Il s'agit de couper le lien avec sa classe actuelle et de l'attacher à l'une des super-classes si cet attachement n'est pas déjà déductible. Il faut ajouter un certain nombre d'assertions dans B'_+ : le second terme de l'union est identique à celui du langage simple; le troisième terme de l'union permet de tenir compte de la perte d'assertion du type $(o \in c)$ que le déplacement peut entraîner dans la clôture déductive de la base. En effet moins de contraintes pèsent sur les valeurs des attributs de l'instance déplacée et donc l'application de la règle de ($\in$-inférence) est restreint. Il s'agit donc d'ajouter les assertions $(o' \in c'')$ telles que :

- cette assertion est déductible grâce à $(o \in c)$ et pas grâce à d'autres instances qui la référencent,
- cette assertion n'est pas déductible grâce à $(o \in c')$,
- cette assertion doit appartenir à la forme première de la base révisée.

pré-condition : c' est la plus basse super-classe de c acceptable pour o.

DI(o, c, c') :

$B'_- = \{o \in c\}$ et

$B'_+ = \{o \in c'$ si E-réf(o,c')=Ø

et s'il n'existe pas c'' telle que $(c'' \leq^* c') \in \text{FP}(B)$ et $(o \in c'') \in \text{FP}(B)\}$

$\cup \{o.a=v$ tel que $\{v\}$=Rest(o,a,Ø) et $\{v\} \neq$Rest(o,a,$[c\ c'[)\}$

$\cup \{o' \in c_a$ si pour tout $(o'',a) \in$ E-réf(o',c_a), $o''=o$ et qu'il existe au moins un tel couple (o,a),

il n'existe pas c_a telle que $(c_a' \leq^* c_a) \in \text{FP}(B)$ et $(c'.a \subseteq^* c_a') \in \text{FP}(B)$

et pour $(c'_a \leq^* c_a) \in \text{FP}(B)$ et $c'_a \neq c_a$, E-réf(o',c'_a)=Ø et $(o' \in c'_a) \notin \text{FP}(B)\}$

+ S-Référence2(o,c,c')

(S-Référence2) se charge de résoudre les inconsistances avec les instances qui référencent l'instance déplacée (Figure 52).

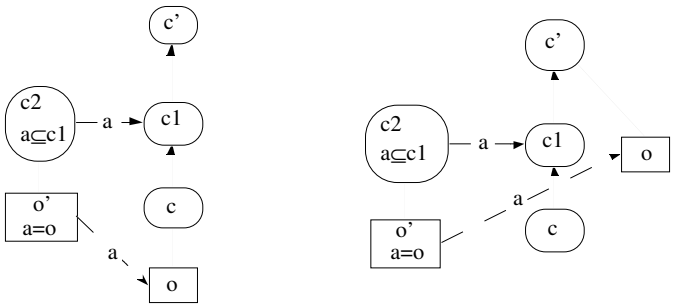


Figure 52 : Pour que le déplacement d'instance soit consistant, il faut résoudre les problèmes sur les instances qui référencent l'instance déplacée. Dans l'exemple, il faut que $(o \in c_1)$ ne soit plus déductible.

Plus formellement, l'ensemble des instances qui posent problèmes sont :

I={$o' \in O$ tel que $(o'.a_1.a_2.\ldots a_n = {}^*o) \in \mathrm{FP}(B)$, $(o' \in c_1) \in \mathrm{FP}(B)$, $(c_1.a_1.a_2.\ldots a_n \subseteq {}^*c_2) \in \mathrm{FP}(B)$, $c_2 \in]c\ c'[$}

Cet ensemble d'inconsistances ressemble fortement à celui exprimé au paragraphe c, à la différence près que les inconsistances concernent ici les instances qui référencent, alors que précédemment les inconsistances portaient sur les instances référencées. La solution adoptée a donc une structure similaire.

Pour résoudre cet ensemble *I* d'inconsistances, il est possible de :

- couper la chaîne $(o'.a_1.a_2.\ldots a_n = {}^*o)$ en supprimant l'une des assertions qui la compose,
- déplacer $(o' \in c_1)$ de la classe c_1 sous la super-classe la plus spécifique qui peut l'accueillir,
- couper la chaîne $(c_1.a_1.a_2.\ldots a_n \subseteq {}^*c_2)$ en modifiant une référence d'attribut de classe vers une classe supérieure.

La modification élémentaire consiste à proposer tout d'abord la suppression des assertions de la forme $(o'.a_i = o)$, puis celle de la forme $(c_a.a_i \subseteq {}^*c'_a)$ en parcourant la chaîne des attributs. De plus, l'ensemble CNC indique l'ensemble des classes d'appartenance qu'il ne faut pas préserver pour une instance *o* lors de la suppression de $(o'.a_i = o)$.

S-Référence2(o,c,c') = S-Réf2-o(o,[$c\ c'$[) + S-Réf2-c(o,c)

S-Réf2-o(o,CNC) :
Pour chaque a_i

Pour chaque o' non marqué tel que $o'.a_i = o$

S(o',a_i,o,CNC) et marquage de l'instance résolue

ou rien sur cette instance et les instances qui la référencent

ou S-Réf2-o(o',{c''' t.q. $(o' \in c'') \in \mathrm{Cn}(B)$ et il existe a_i t.q. $(c'''.a_i \subseteq {}^*c'') \in \mathrm{CNC}$)

S-Réf2-c(o,c) :

Pour chaque (a_i,c_i) tels qu'il existe $(o'.a_i = o) \in \mathrm{FP}(B)$, $(o' \in ci) \in \mathrm{Cn}(B)$, o' non marqué $c_i.a_i.a_{i+1}.\ldots a_n \subseteq {}^*c'_a$ et $c'_a \leq c'$:

ARAC(c_i,a_i,c'_a) avec c'_a telle que $(c_a \leq {}^*c'_a) \in \mathrm{FP}(B)$, $c'_a.a_{i+1}.\ldots a_n \subseteq {}^*c''$, $(c' \leq {}^*c'') \in \mathrm{FP}(B)$ et pour c''_a telle que $c''_a \in [c_a\ c'_a[$, $c''_a.a_{i+1}.\ldots a_n \subseteq {}^*c'''$ et $\neg(c''' \leq {}^*c'') \in \mathrm{FP}(B)$ et on supprime de B'_- les $(o.a = o')$ qui ont été supprimées inutilement

ou si $(o' \in c) \notin \mathrm{FP}(B)$, S-Réf2-c($o',c_i$)

e) Déplacer une classe sous une super-classe

Il s'agit de couper le lien de spécialisation entre deux classes et d'en créer un avec l'une des super-classe; de plus, il s'agit de préserver toutes les restrictions d'attributs possibles. L'ensemble d'attributs *ANC* (Attributs Non Conservés) indique les attributs à cause desquels le déplacement de classes est effectué et donc les attributs dont il ne faut pas préserver les restrictions. Pour les attributs pour lesquels on veut préserver l'ancienne restriction, il s'agit de conserver la restriction qui apparaissait dans la forme normale, et qui correspond à celle de la restriction de la super-classe la plus spécifique.

pré-condition : c_m est la super-classe de c_i la plus spécifique qui puisse être sa super-classe directe (c'est-à-dire telle que l'assertion ($c_i \leq c_m$) soit acceptable) si l'on considère les attributs n'appartenant pas à ANC.

DC(c_i, c_m, *ANC*):

$B'_- = \{c_i \leq c_{i+1}\}$

$B'_+ = \{c_i \leq c_m\}$

$\cup \{c_i.b \subseteq d$, si $(c_i.b \subseteq d) \in \mathrm{FN}(B)$ et $(c_m.b \subseteq d) \notin \mathrm{FN}(B)\}_{b \notin ANC}$

$\cup \{o.a = v$ tel que $(o \in c_i) \in \mathrm{Cn}(B)$, $\{v\} = \mathrm{Rest}(o,a,\emptyset)$ et $\{v\} \neq \mathrm{Rest}(o,a,]c_i\ c_m[)\}_{a \in ANC}$

Deux problèmes se posent alors : résoudre le problème pour les instances (est-ce qu'elles restent attachées à la classe déplacée ou attachées à une super-classe ?) et gérer les inconsistances soulevées dans les classes qui référencent les classes déplacées.

(1) Problème des instances

Lorsqu'une classe est déplacée, il existe deux possibilités pour ses instances et les instances de ses sous-classes :

(1) soit elles suivent la classe déplacée et le rattachement aux classes comprises entre la super-classe initiale et la super-classe finale est perdu,
(2) soit elles sont rattachées à la super-classe de la classe déplacée, mais à ce moment-là le rattachement de l'instance aux sous-classes de la classe déplacée est perdu (cet ensemble d'instances sera appelé ID).

Dans le cas où plusieurs classes sont déplacées (par exemple dans le cas traité en a.), l'ensemble des classes déplacées sera noté CD. La première alternative n'est pas modifiée, alors que la deuxième revient à déplacer l'instance sous la super-classe la plus spécifique de sa classe d'appartenance qui n'a pas été déplacée.

La première alternative ne nécessite aucune modification directe de la base, alors que la seconde nécessite les modifications décrites ci-dessous :

Dans le cas où seule la classe c est déplacée,

$RI(c) = \{DI(o,c_k,c') \text{ si } \exists k \text{ tel que } (c_k \leq^* c) \in FP(B), (o \in c_k) \in FP(B),$
$\text{et } c' \text{ est tel que } (c \leq c') \in FP(B)\}$

Plus généralement, dans le cas où l'ensemble de classes déplacées est CD :

$RI(CD) = \{DI(o,c_k,c') \text{ si } \exists\, c \in CD,\ k \text{ tel que } (c_k \leq^* c) \in FP(B), (o \in c_k) \in FP(B), \text{ et } c' \text{ est la plus spécifique super-classe de } c \text{ non déplacée}\}$

Si la solution globale impose que certaines instances ne doivent pas être rester attachées à leur classe de rattachement initiale, elles seront spécifiées de la manière suivante : $RI(c_i) \backslash \{o\}$, $\{o\}$ étant l'ensemble d'instances à ne pas déplacer.

(2) Problème des références

Par rapport au cas du langage simple, des problèmes apparaissent dans les classes qui référencent des classes déplacées. Rappelons qu'une classe est incohérente lorsque l'intersection des restrictions posées sur un attribut est l'ensemble vide. Lorsqu'il s'agit de restriction d'attribut par des classes, une classe est incohérente si l'intersection des classes référencées est vide.

Lors du déplacement d'une classe, des classes incohérentes peuvent donc apparaître, et ce dans diverses hiérarchies. Il semble difficile d'imaginer une résolution indépendante pour les classes d'une hiérarchie donnée, vu que le concepteur devrait intervenir à chaque choix et qu'il risquerait de faire des modifications inutiles ou contradictoires. C'est pourquoi, une résolution globale – sur chaque hiérarchie de classes – est proposée pour résoudre les inconsistances soulevées par ce déplacement de classe. Les résolutions sur les différentes hiérarchies sont supposées indépendantes, même si une résolution sur une hiérarchie peut elle-même influer sur d'autres hiérarchies. Rappelons toutefois que, par hypothèse, les références entre classes n'admettent pas de cycles, et par conséquent que l'algorithme qui va résoudre les incohérences en parcourant les hiérarchies ne va pas boucler.

Ci-dessous nous détaillons le cas d'une hiérarchie qui pose problème pour l'un de ses attributs. Rappelons tout d'abord que l'incohérence potentielle provient du déplacement de classes représenté dans la Figure 53.

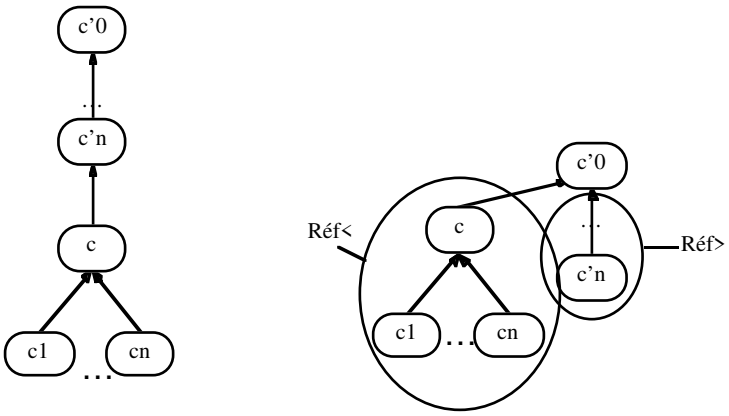


Figure 53 : Un déplacement de classes, ainsi que les ensembles Réf< et Réf>, sont représentés

On note Réf>, l'ensemble des classes de la branche entre la super-classe de la classe déplacée et la sous-classe de la classe sous laquelle est attachée la classe déplacée : ce sont les super-classes de la classe déplacée qui ne le sont plus après déplacement. On note Réf<, l'ensemble des classes déplacées. Ces deux ensembles vont permettre de déterminer les classes problématiques dans les hiérarchies qui les référencent. Les ensembles de classes dont l'attribut *a* référence une classe de Réf> ou Réf< sont notés respectivement Réf>(a) et Réf<(a) (Figure 54).

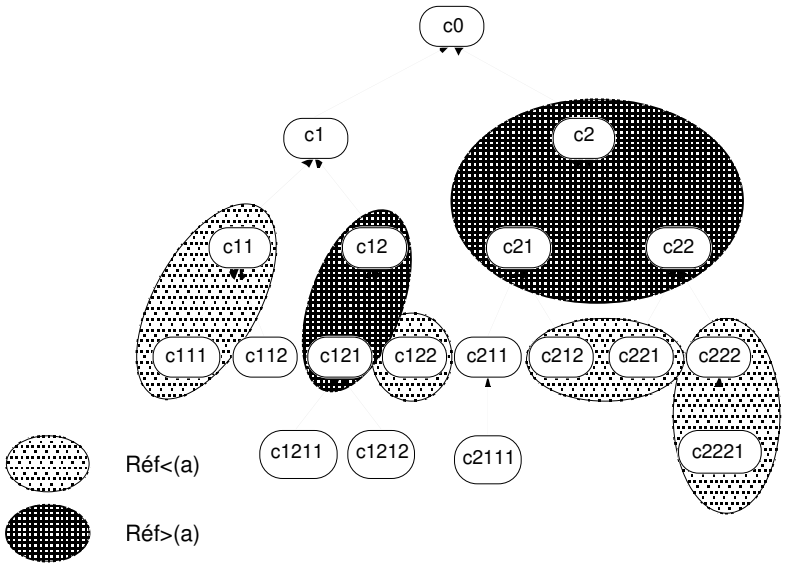


Figure 54 : Sur la hiérarchie de classes ont été indiquées les classes dont l'attribut *a* référence une classe de Réf> ou une classe de Réf<; ces ensembles étant définis par rapport au déplacement de classes.

Il y a une incohérence lorsqu'une classe est dans Réf<(*a*) et que l'une de ses super-classe est dans Réf>(*a*); en effet, la base était acceptable avant le déplacement de classe, mais après celui-ci, la sous-classe est incohérente, car l'attribut *a* considéré est restreint par deux classes

qui ne sont plus liées par la relation de spécialisation. Les ensembles Réf>(*a*) et Réf<(*a*) peuvent donc être réduits, pour ne garder que les éléments qui posent problème, c'est-à-dire les couples de classes dont l'une est dans Réf<(*a*) et l'autre dans Réf>(*a*). Pour toutes les autres classes, aucune incohérence n'est possible. Ainsi, *c*11 et *c*111 ne posent en fait pas de problème particulier, vu que :

(1) la relation de spécialisation est vérifiée entre ces deux classes car elle l'était avant le déplacement de classe (car la base est acceptable initialement) et que les classes que référencent leur attribut *a* n'ont pas été déplacées l'une par rapport à l'autre.

(2) la relation de spécialisation est vérifiée par rapport à leurs super-classes (c1 par exemple) qui référencent des super-classes de la classe sous laquelle a été déplacée la classe car elles ne peuvent pas référencer des sous-classes (sinon leurs super-classes appartiendraient à Réf<(*a*) ou à Réf>(*a*) — ce qui n'est pas le cas — ou elles référenceraient des sous-classes de classes de Réf< — ce qui n'est pas possible sinon il y aurait incohérence avant le déplacement de classe).

De même tous les éléments de Réf<(*a*) dont aucune super-classe n'appartient à Réf>(*a*) et tous les éléments de Réf>(*a*) dont aucune sous-classe n'appartient à Réf<(*a*) (par exemple *c*121)ne sont pas à prendre en considération. Sur l'exemple traité, seul l'ensemble de classes réellement problématiques est conservé (Figure 55).

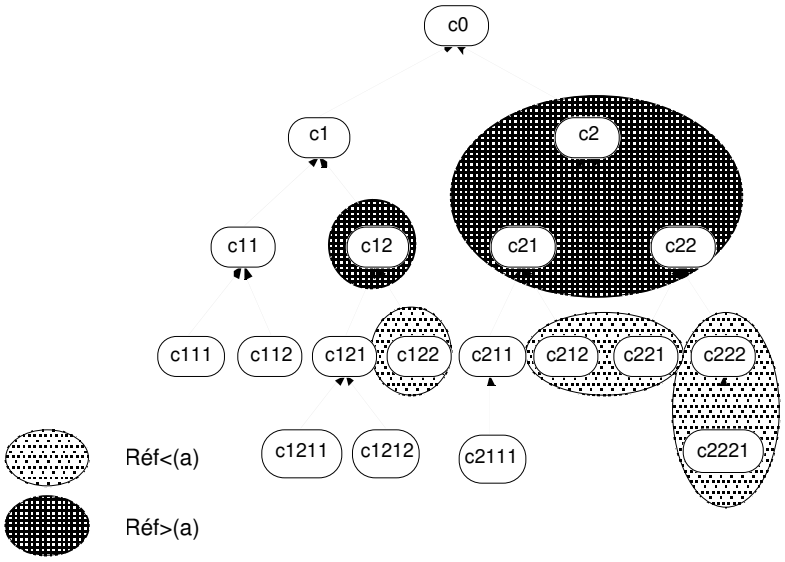


Figure 55 : Dans la hiérarchie de classes de la Figure 54, seules les classes problématiques sont encore représentées dans les ensembles Réf>(*a*) et Réf<(*a*).

Propriété :

Quelle que soit c telle que : $c1 \leq c \leq c2$, avec $c1 \in$ Réf<(a) et $c2 \in$ Réf>(a), alors c appartient à Réf<(a) ou à Réf>(a) (informellement, il n'existe pas de "trous" entre Réf<(a) et Réf>(a)).

Pour réviser, il faut traiter le problème pour toutes les classes de Réf<(a) et Réf>(a). Nous allons envisager une solution globale, c'est-à-dire qui traite le problème pour toutes les classes d'un ensemble connexe de Réf>(a) et de ses sous-classes de Réf<(a). Le déplacement des classes impliquées dans le déplacement de classe initial n'est pas envisagé car on ne veut pas remettre en cause les modifications précédentes. Les différentes solutions élémentaires possibles sont :

(1) modifier les attributs des classes appartenant à Réf>(a) pour qu'ils référencent la classe $c'0$ (qui est la nouvelle super-classe de la classe déplacée).
(2) modifier les attribut des classes appartenant à Réf<(a) pour qu'ils référencent la classe référencée par leur super-classe.
(3) déplacer les classes de Réf<(a) (ou de Réf>(a) dont la référence a été modifiée) sous des super-classes n'appartenant pas à Réf>(a) ou appartenant à Réf>(a) et dont la référence a été modifiée.

On ne veut envisager que les solutions qui font des modifications minimales; ainsi, il ne faut pas modifier une référence d'une classe de Réf>(a) et déplacer toutes ses sous-classes appartenant à Réf<(a) (sinon le changement de référence est inutile)

	la solution …	résout les incohérences pour …
(1)	modifier référence d'une classe de Réf>(a)	ses sous-classes de Réf<(a) à la condition que les références soient déplacées également dans ses sous-classes de Réf>(a).
(2)	modifier référence d'une classe de Réf<(a)	elle-même
(3)	déplacer une classe de Réf>(a) déplacer une classe de Réf<(a)	elle-même elle-même et ses super-classes de Réf>(a) si les incohérences sont résolues dans toutes les sous-classes.

L'algorithme est le suivant :

1. Parmi l'ensemble de classes Réf>(a) et Réf<(a), choisir les classes qui sont déplacées (3) en préservant au maximum les liens de spécialisation, c'est-à-dire qu'une classe est déplacée sous son ancienne super-classe déplacée la plus spécifique. Pour les classes de

Réf>(*a*), ce déplacement s'accompagne d'une modification des restrictions qui pèsent sur l'attribut *a* (1). Une contrainte supplémentaire est posée : pour chaque classe de Réf>(*a*) qui est déplacée, il existe au moins une de ses sous-classes appartenant à Réf<(*a*) qui est déplacée ou dont la restriction n'a pas besoin d'être modifiée (cela évite de faire une modification inutile dans une classe de Réf>(*a*))

2. Parmi les classes de Réf<(*a*), celles dont toutes les super-classes appartenant à Réf>(*a*) ont été déplacées ne sont plus inconsistantes. La restriction de l'attribut *a* est modifiée pour les autres (2).

Les classes explicitement déplacées ou les classes de Réf<(*a*) qui ne sont plus inconsistantes après l'étape 1 forment l'ensemble CD.

Les bases révisées obtenues par un tel algorithme sont minimales car aucune modification n'est superflue. Un exemple est donné par la Figure 56.

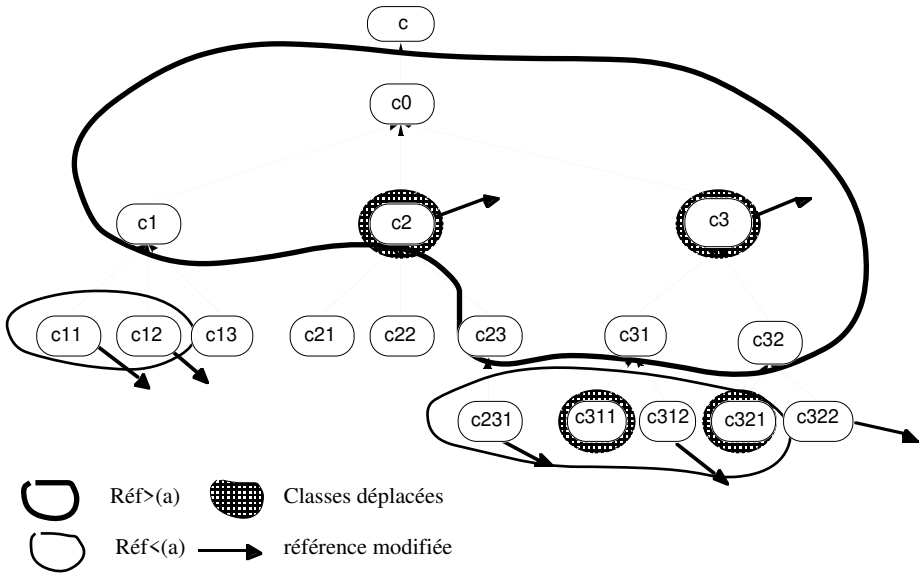


Figure 56: exemple d'application de l'algorithme décrit ci-dessus : on choisit de déplacer certaines classes, les références des classes de Réf<(a) étant alors modifiées pour celles qui le nécessitent.

Les classes c2, c3, c311 et c321 sont déplacées et les références des classes c2 et c3 modifiées. Le problème est alors résolu pour les classes c21 et c22 et pour les autres, il faut modifier leur référence. Le résultat de cette modification de la base est illustré par la Figure 57. Remarquer que la relation de spécialisation est conservée entre c321 et c3 par exemple.

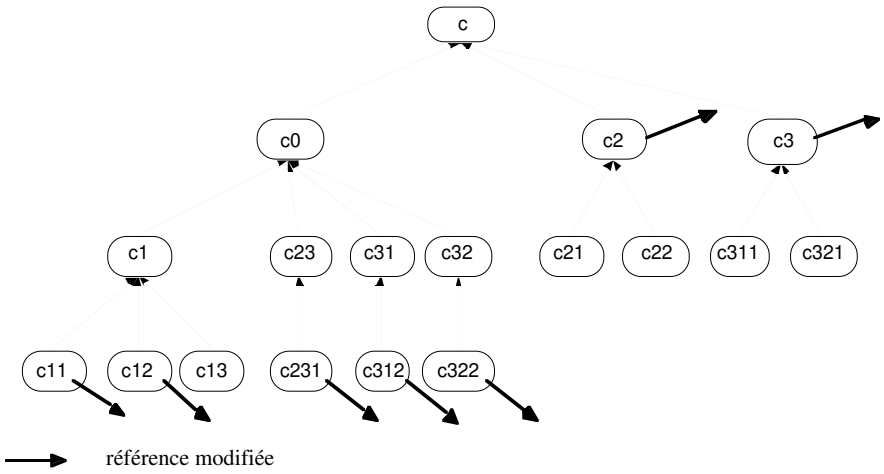


Figure 57: la hiérarchie de classe après application de l'algorithme.

Le problème des classes incohérentes est donc résolu lorsqu'*un* attribut d'une hiérarchie de classes référence des classes dans une hiérarchie dans laquelle une classe a été déplacée. Cependant, il est possible que *plusieurs* attributs référencent des classes dans cette hiérarchie. L'obtention d'une base révisée minimale est alors plus délicate. Les solutions élémentaires (1) et (2) sont indépendantes pour chaque attribut; par contre la solution (3) (le déplacement de classes) ne peut s'envisager séparément pour les attributs problématiques. L'algorithme est alors modifié de la façon suivante :

1. Choisir les classes déplacées (ensemble noté CD) avec la contrainte suivante : pour toute classe déplacée telle que $c \in \text{Réf}>(a_i)$ pour $a_i \in A_i$, pour chaque a_i, il existe une classe c' telle que $c' \leq^* c$, $c' \in \text{Réf}<(a_i)$ et c' est déplacée.
2. Modifier les références pour l'attribut a_i pour les classes $c \in \text{Réf}<(a_i)$ s'il existe au moins une de ses super-classes qui n'a pas été déplacée.

3. Solution globale

Pour définir les bases révisées, il sera utile de savoir localiser d'où vient le problème et en particulier de connaître la classe dans laquelle est posée la restriction d'attribut problématique; en fait, il s'agit de la classe la plus spécifique ayant une définition de restriction d'attribut, vu la définition de la forme première. En effet, si un domaine qui restreint un attribut dans une classe ne peut contenir un autre domaine, il en sera de même de tous les domaines de restriction dans les sous-classes, et inversement si un domaine qui restreint attribut peut contenir un domaine quelconque, ce domaine sera aussi inclus dans les restrictions d'attribut de toutes les super-classes. Cette classe qui contient la première restriction problématique est définie formellement ci-dessous :

DÉFINITION 35 : PREMIÈRE CLASSE INCONSISTANTE (AVEC UNE RESTRICTION PAR UNE CLASSE)

PCI(c,a,c') est la classe la plus spécifique qui est super-classe de c et qui n'est pas compatible avec la classe c' pour l'attribut a :

$PCI(c,a,d) = c_n$ t.q. $FP(B) \vdash c \leq c_n$, $(c_n.a \subseteq d_n) \in FP(B)$ et $\neg FP(B) \vdash c' \leq d_n$.

DÉFINITION 36 : PREMIÈRE CLASSE INCONSISTANTE (AVEC UNE INSTANCE)

PCI(c,a,$\{o\}$) est la classe la plus spécifique qui est super-classe de c et qui n'est pas compatible avec l'instance o pour l'attribut a :

$PCI(c,a,\{o\}) = c_n$ t.q. $FP(B) \vdash c \leq c_n$, $(c_n.a \subseteq c'_n) \in FP(B)$ et $\neg FP(B) \vdash o \in c'_n$.

4. L'assertion ($o.a=v$)

En utilisant les solutions élémentaires adaptées au cas du langage complexe, le traitement de cette assertion est identique au cas du langage simple.

5. L'assertion ($o.a=o'$)

a) Problèmes soulevés

La principale différence avec le cas du langage simple est que lorsque $B \vdash o \in c$, cette assertion peut avoir été déduite par ($\in$-inférence) et donc si l'on décide de supprimer l'assertion ($o \in c$), cela sera plus complexe. De plus, l'inconsistance peut porter sur une instance que l'instance référence par le biais d'une chaîne d'attributs. Reprenons un exemple donné au paragraphe III.A.2.

Les seuls problèmes possibles sont du type (1) et (2) :

(1) $(o.a=o'') \in FP(B)$ avec $o'' \neq o''$

(2) $(o \in c)$ et il existe $c.a.a_1.a_2.\ldots.a_n \subseteq *d'$, $v.a_1.a_2.\ldots.a_n = *v'$ et $\neg v' : d'$

La Figure 58 donne un exemple d'inconsistance du type (2). Il peut exister plusieurs classes et plusieurs chaînes d'attributs pour lesquelles il y a une inconsistance du type (2).

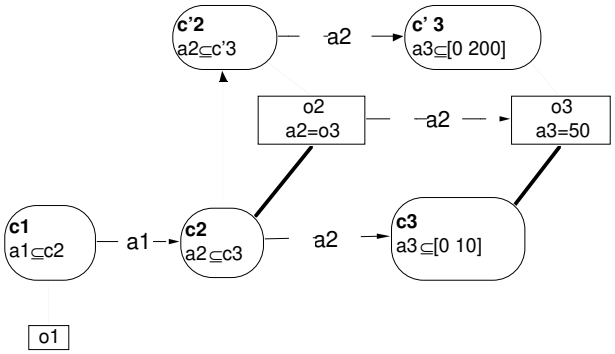


Figure 58: Si on ajoute *o1.a1=o2* dans la base, la règle de (∈-inférence) permet de déduire (*o2*∈*c2*) et donc (*o3*∈*c3*), d'où inconsistance d'après le point (2) entre la valeur de *o3.a3* (50) et les restrictions posées sur cet attribut dans sa nouvelle classe d'appartenance ([0 10])

b) Solutions élémentaires

Pour résoudre (1)

(solution 1) – *supprimer la valeur de l'attribut*

Si $(o.a=v') \in FP(B)$, alors $S1=S(o,a,v')$

Pour résoudre (2) :

Pour chacune des classes d'appartenance *c* de *o*, il est possible de déplacer *o* ou de traiter le problème pour l'attribut *a* de *o*, d'où

(solution 2) – déplacer l'instance

$DI(o,c,c')$ avec c' la super-classe la plus spécifique de *c* telle que si $c'.a.a_1.a_2.\ldots.a_n \subseteq *d''$, $v.a_1.a_2.\ldots.a_n = *v'$ alors $v':d''$.

(solution 3) – *supprimer les références à un objet*

S-Référence1(*o,a,o',c*)

c) Solution globale

La solution est composée des différentes solutions élémentaires.

1. $S=\{ B'_- = \emptyset;\ B'_+ = \emptyset \}$

2. Si $(o.a=v') \in FP(B)$, $S=S1$

3. Si des inconsistances du type (2) existent, pour chaque classe d'appartenance de *o* S=S+(S2 ou S3)

6. L'assertion ($o \in c$)

a) Problèmes soulevés

Les nouvelles inférences possibles sont les assertions indiquant de nouvelles valeurs pour des attributs de *o*(ayant une valeur de type) et de nouvelles classes d'appartenance pour des instances valeurs des attributs de *o*. Les inconsistances possibles sont alors du second type ou du troisième type. La valeur atteinte par une chaîne d'attribut à partir de *o* peut n'être pas compatible avec les restrictions posées sur la classe référencée par cette même chaîne d'attributs (2) (Figure 59). De plus, *o* qui appartient à une nouvelle classe peut appartenir à présent à un ensemble de classes incompatibles, de même pour les instances référencées par *o* (3) (Figure 60).

(2) $\{ni \geq 1,\ o.a_1.a_2.\ldots.a_{ni}=v_i,\ (c.a_1.a_2.\ldots.a_{ni} \subseteq *d_i) \in \mathrm{FP}(B)$ et $\neg v_i:d_i\}$

(3) $\{ni \geq 0,\ o.a_1.a_2.\ldots.a_{ni}=*o_i,\ (c.a_1.\ldots.a_{n'i} \subseteq *d_i) \in \mathrm{FP}(B)$ et $B \vdash o_i \in c_i,\ (c_i.a_{ni+1}.\ldots.a_{n'i} \subseteq *d_{ij}) \in B,$

et $\underset{ij}{\wedge} d_{ij} \wedge d_i = \perp\}$

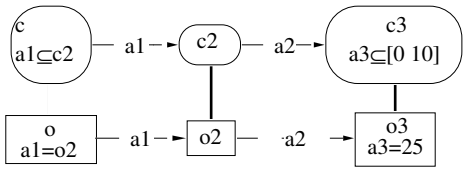


Figure 59: Cette figure explicite l'un des cas possible d'inconsistance; les traits en gras indiquent les rattachement d'instance qui sont inférés par la règle de ($\in$-inférence). Il est possible d'inférer qu'une instance ne satisfait pas les contraintes d'une nouvelle classe d'appartenance.

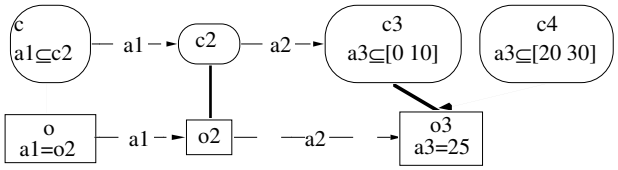


Figure 60 : Cette figure illustre le cas (3) d'inconsistance; l'instance o3 est référencée par *o* et elle donc attachée à *c3* par la règle de ($\in$-inférence); elle appartient donc à deux classes qui sont incompatibles.

b) Solutions élémentaires

Les inconsistances du type (2) peuvent être résolues de diverses manières :

- déplacer la classe *c* si elle n'admet pas de restrictions d'attribut dans la forme première;
- couper la chaîne d'instance ($o.a_1.a_2.\ldots.a_n=v$) en supprimant l'une des assertions du type ($o_i.a_i=o_{i+1}$);
- modifier des restrictions d'attribut des classes intervenant dans la chaîne ($c.a_1.a_2.\ldots.a_n \subseteq *d_i$) : ($c_i.a_{i+1} \subseteq c_{i+1}$) remplacée par ($c_i.a_{i+1} \subseteq c'_{i+1}$) avec c'_{i+1} super-classe de c_{i+1}.

Les inconsistances du type (3) peuvent quant à elles être résolues par :

- couper la chaîne d'instance ($o.a_1.a_2.\ldots.a_n=v$) en supprimant l'une des assertions du type ($o_i.a_i=o_{i+1}$);
- modifier les autres classes d'appartenance de o_i, une instance référencée par *o*
- modifier des restrictions d'attribut des classes intervenant dans la chaîne ($c.a_1.\ldots.a_{n'i} \subseteq *d_{ij}) \in B$: ($c_i.a_{i+1} \subseteq c_{i+1}$) remplacée par ($c_i.a_{i+1} \subseteq c'_{i+1}$) avec c'_{i+1} super-classe de c_{i+1}.

- modifier des restrictions d'attribut des classes intervenant dans les chaînes $(c_i.a_{ni+1}.\ldots.a_{n'i} \subseteq *d_{ij}) \in B$, avec $B \vdash o \in c_j$: $(c_i.a_{i+1} \subseteq c_{i+1})$ remplacée par $(c_i.a_{i+1} \subseteq c'_{i+1})$ avec c'_{i+1} super-classe de c_{i+1}.

Ces différentes stratégies ne sont pas indépendantes pour les différents ensembles d'attributs qui posent problème.

Pour résoudre (2) et (3) si l'inconsistance ne porte pas sur *o*

(solution 1') – *déplacer la classe*

Le déplacement de la classe est identique à celui effectué dans le cas du langage simple, à la différence près que les ensembles PCP et PCI sont définis différemment.

(solution 2) – *supprimer la valeur de l'attribut*

S($o.a=o'$)

(solution 3) – *modifier les instances référencées par l'instance*

S-réf1-o(o')

Pour résoudre (2) et (3) quelle que soit l'instance

(solution 4) – *modifier les classes référencées*

S-référence1(o,a,o',c)

La valeur qui est ajoutée dans les domaines pour résoudre (2) est la valeur qui apparaît dans l'inconsistance et pour résoudre (3) c'est l'intersection des domaines des autres classes d'appartenance.

Pour résoudre (3)

(solution 5) – *modifier les classes référencées des anciennes classes d'appartenance*

S-réf1-c(c_i,a_{1i}) pour chaque classe d'appartenance de l'instance.

La valeur qui est ajoutée dans les domaines est le domaine de la nouvelle classe d'appartenance.

(solution 6) – *déplacer l'instance de ses anciennes classes d'appartenance*

L'instance doit peut-être conserver certaines de ses classes d'appartenance, c'est pourquoi, on propose tout d'abord de choisir parmi toutes les classes celles qui ne seront plus classe d'appartenance; soient C-depl et C-inch, respectivement les ensembles de classes d'appartenance qui ne le seront plus et celles qui sont conservées.

Les nouvelles classes sont les super-classes qui ont des restrictions d'attributs conformes avec l'intersection du domaine de la nouvelle classe d'appartenance et des anciennes qui sont conservées.

Pour toute classe de C-depl

DI(o_i', c_i, c_i') avec c_i' la super-classe la plus spécifique de c telle que $(c'_i.a_{ni+1}.\ldots a_{n'i} \subseteq * d'_{ij}) \in B$, et $\underset{d_{ij} \text{ domaines des classes de C-inch}}{\wedge} d_{ij} \wedge d_i \subseteq d'_{ij}$

c) Solution globale

1. S'il y a des inconsistances concernant *o* directement :
   S=S6 pour résoudre un ensemble d'attributs problématiques, puis pour tous ceux qui le sont encore : S=S+(S4 ou S5)
2. S'il y des inconsistances faisant intervenir des instances référencées par *o* :
   S=S+S1', puis pour chaque attribut problématique S=S+(S2 ou S3)
3. Pour chaque inconsistance du type (3) qui n'a pas été résolue
   S=S+(S5 ou S6)

7. L'assertion ($c \leq c'$)

De même que précédemment la migration de classes n'est pas autorisée. Par rapport au langage simple, le déplacement sous une super-classe peut soulever inconsistance et/ou incohérence à cause des références à des classes. Ces problèmes sont résolus par les solutions données dans le paragraphe 2.e.

Si l'hypothèse n'est pas posée, la résolution est similaire que celle pour le langage simple.

8. L'assertion ($c.a \subseteq d$), $d \in T$

Les problèmes et la résolution sont identiques à précédemment, grâce aux modifications élémentaires qui ont été modifiées pour tenir compte des références aux instances (en particulier le déplacement de classes tient compte des classes qui les référencent).

9. L'assertion ($c.a \subseteq d$), $d \in C$

a) Problèmes soulevés

Si $d \in C$, les problèmes pouvant apparaître sont les mêmes, sauf qu'ils peuvent être localisés dans une hiérarchie éloignée de celle dans laquelle est fait l'ajout. En utilisant les modifications élémentaires qui tiennent compte des références aux instances, la résolution est quasiment identique.

Les problèmes pouvant apparaître sont des classes incohérentes et des instances inconsistantes dans tout le sous-arbre de la classe modifiée. La classe c est incohérente s'il existe une restriction dans l'une de ses super-classes telles que les deux classes référencées ne soient pas compatibles. Ces sous-classes de c sont incohérentes si leur attribut est restreint par une classe non compatible avec c. Soit C l'ensemble des incohérences.

$C=\{c$ si $\exists c_i$ $(c_i.a \subseteq d_i) \in \mathrm{FP}(B)$, $(c \leq^* c_i) \in \mathrm{FP}(B)$ et il existe $a_1, a_2, \ldots, a_n$ tels que $(d_i.a_1.a_2.\ldots.a_n \subseteq^* d_i') \in \mathrm{FP}(B)$, $(d.a_1.a_2.\ldots.a_n \subseteq^* d') \in \mathrm{FP}(B)$ et $d' \wedge d_i' = \bot\}$

$\cup \{c_i$ t.q. $(c_i.a \subseteq d_i) \in \mathrm{FP}(B)$, $(c_i \leq^* c) \in \mathrm{FP}(B)$ et il existe $a_1, a_2, \ldots, a_n$ tels que $(d_i.a_1.a_2.\ldots.a_n \subseteq^* d_i') \in \mathrm{FP}(B)$, $(d.a_1.a_2.\ldots.a_n \subseteq^* d') \in \mathrm{FP}(B)$ et $d' \wedge d_i' = \bot d\}$

L'ensemble des inconsistances contient alors les instances attachés à une classe incohérente ou celles dont la valeur d'attribut n'est pas compatible avec la nouvelle restriction :

$I=I1 \cup I2$ avec
$I1=\{o, B \vdash o \in c_i, c_i \in C\}$ et
$I2=\{o, B \vdash o \in c, (o.a.a_1.a_2.\ldots.a_n = *v') \in \mathrm{FP}(B), (d.a_1.a_2.\ldots.a_n \subseteq^* d') \in \mathrm{FP}(B)$ et $\neg v' : d'\}$

b) Solutions élémentaires

Les solutions élémentaires sont les mêmes que celles définies dans le cas simple pour l'assertion $(c.a \subseteq d)$, à la seule différence près que la solution ARAC sera utilisée à la place de la solution ARA.

De plus, dans le cas de déplacement de classes, comme il est possible que plusieurs classes soient déplacées, la solution préconisée pour traiter les problèmes liés aux références devra être appliquée en tenant compte de l'ensemble des déplacements, c'est-à-dire que l'ensemble Réf< sera composé uniquement des classes qui ont été déplacées (l'ensemble Réf> étant identique à celui précédemment défini).

c) Solution globale

La solution globale est également très fortement similaire à la solution globale dans le cas du langage simple. Nous la rappelons ci-dessous et donnons les points particuliers qui ont été modifiés

1. Si $c \in C$, S=(S1 ou S2 ou S2' ou S3) (C <–$C \backslash \{c\}$)

2. Pour $c_i \in C$, telle qu'elle n'a plus de super-classe dans C :

   S=S+S4
   ou S=S+S5
   ou S=S+S6

3. Répéter 2 tant que $C \neq \emptyset$

4. Pour $o \in I2$, S=S+(S7 ou S8); de plus à chaque suppression d'une assertion du type ($o.a=v$), on enlève de $B'_$, celles qu'il n'est plus nécessaire de supprimer et on marque celles qui sont résolues. La mise en œuvre concrète nécessite de parcourir les chemins d'attribut qui contiennent l'assertion supprimée afin de voir si les assertions précédemment supprimées sont encore nécessaires (Figure 61 et Figure 62).
5. Les problèmes liés aux déplacements de classes sont résolus selon les solutions données au paragraphe 2.e.2.

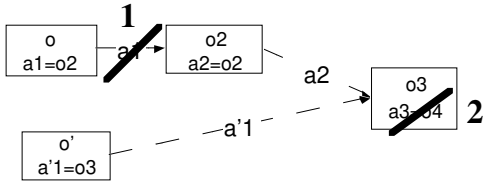


Figure 61 : illustration du point 4 de l'algorithme : soient o et o' les deux instances problématiques et les attributs représentés les chemins problématiques ; supposons que l'on supprime en premier lieu $o.a_1=o_2$; le suppression dans un second temps de $o_3.a_3=o'$ rend inutile la première suppression : l'assertion $o.a_1=o_2$; doit être conservée dans la base révisée. Dans le cas où 2 est effectué en premier, le fait que l'inconsistance est résolue est propagée pour ne pas faire de suppression inutile.

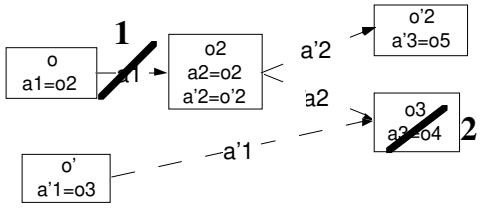


Figure 62 : illustration de la vérification nécessaire qui doit tenir compte de tous les chemins d'attributs problématiques d'un objet : les deux suppressions effectuées sont nécessaires dans ce cas.

D. Des alternatives à la révision

Suite à l'ajout d'une connaissance inacceptable dans une base, le concepteur peut obtenir les bases révisées minimales qui contiennent la dernière assertion ajoutée et qui préservent le maximum des assertions de la clôture déductive de la base initiale. Or, ces deux contraintes ne sont pas forcément recherchés par l'utilisateur. Il peut en effet vouloir modifier l'assertion qu'il a ajoutée (paragraphe 1) ou obtenir des bases révisées qui ne satisfont pas le critère de minimalité (paragraphe 2).

1. Modifier l'assertion initiale

Lorsque le concepteur ajoute une assertion dans la base, il espère que celle-ci soit acceptable dans la base. Si tel n'est pas le cas, il peut vouloir l'insérer à tout prix et c'est le mécanisme de révision qui s'en charge. Or, voyant que son assertion n'est pas acceptable, il peut également vouloir la retirer ou la modifier. Il semble donc utile de proposer en marge de la révision, un mécanisme aidant le concepteur à modifier son assertion. De plus, pouvoir modifier l'assertion ajoutée permet de limiter l'influence de l'ordre dans lequel les assertions sont ajoutées à la base. Pour chaque assertion, il est donc proposé à l'utilisateur de ne pas réaliser l'ajout d'assertion; de plus, ci-dessous sont détaillées les propositions qui peuvent lui être faites pour la modifier selon son type.

a) L'assertion ($c \leq c'$)

Selon l'hypothèse posée lors de la description de l'algorithme, de telles assertions ne sont autorisées que si la classe c ne possède ni restriction d'attribut, ni instance, ni sous-classe et ce type d'assertions ne soulève donc pas de problème.

b) L'assertion ($o \in c$)

Cette instance ne peut pas être attachée à la classe spécifiée, mais elle peut être attachée sous l'une des super-classes. Il est possible de déterminer la super-classe la plus spécifique qui est une classe de rattachement possible pour o. En fait, on envisage de combiner la modification de l'assertion avec le processus de révision. Il est proposé d'attacher l'instance à une des super-classes de c, ce qui ne résout pas forcément tous les problèmes, ceux-ci étant résolus par le mécanisme de révision. Plus précisément, les inconsistances qui sont déclenchées concernent un ensemble d'attributs; il est possible de déterminer la super-classe la plus spécifique qui résout l'inconsistance pour chaque attribut et de proposer au concepteur le déplacement sous cette classe :

Soit A=$\{a$ tel que $(o.a=v) \in \mathrm{FN}(B)$, $(c.a \subseteq d) \in \mathrm{FN}(B)$ et $\neg v{:}d\}$. On va proposer à l'utilisateur le rattachement à l'une des classes de l'ensemble suivant :

$\{c'$ t.q $(c \leq^* c') \in \mathrm{FP}(B)$, $a \in$ A avec $(o.a=v) \in \mathrm{FN}(B)$, $(c'.a \subseteq d') \in \mathrm{FN}(B)$ et $v{:}d'$

et pour c'' telle que $(c \leq^* c'') \in \mathrm{FP}(B)$ et $(c'' \leq c') \in \mathrm{FP}(B)$, $(c''.a \subseteq d'') \in \mathrm{FN}(B)$ et $\neg v{:}d''\}$

Si $(o \in c')$ est choisi et que cette assertion n'est pas consistante, il faut alors réviser la base.

EXEMPLE :

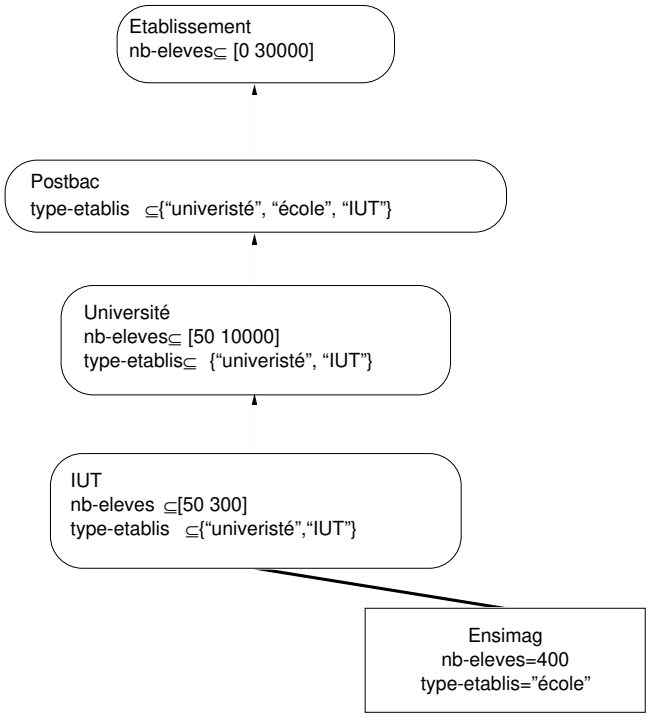


Figure 63 : L'attachement de l'instance `Ensimag` à la classe `IUT` provoque des inconsistances dues aux attributs `nb-eleves` et `type-etablis`.

Consulter la base scolarite

Vous vouliez rattacher l'instance #<< etablissement "Ensimag", "Grenoble" >> a la classe IUT
Inconsistance : l'instance #<< etablissement "Ensimag", "Grenoble" >> ne peut etre attachee a la classe IUT a cause des attributs ("nb-eleves" "type-etablis")

◆Choisir une autre classe

◇Conserver la classe choisie

◇ si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant

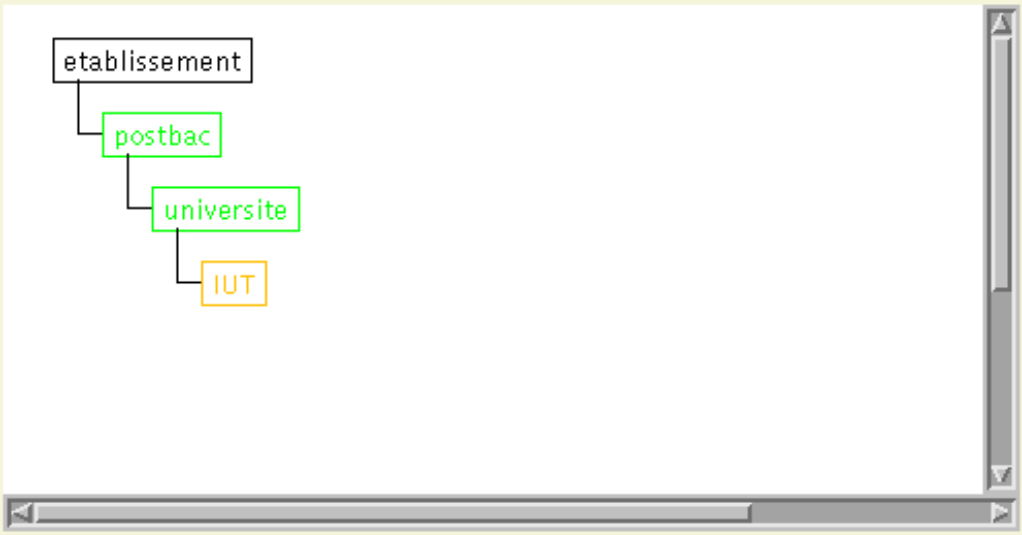
OK

Consulter la base scolarite

Vous vouliez rattacher l'instance #<< etablissement "Ensimag", "Grenoble" >> a la classe IUT
(representee en orange) et cela est problematique pour les attributs ("nb-eleves" "type-etablis")
Vous pouvez choisir une autre classe pour rattacher l'instance, en sachant que cela resout les
inconsistances pour les attributs indiques entre parantheses :
postbac ("nb-eleves" "type-etablis")
universite ("nb-eleves")
IUT ()

Choisissez la nouvelle classe d'appartenance de l'instance. Attention, vous devez choisir une classe
indiquee en vert (eventuellement la classe initiale indiquee en orange)

Note :pour les attributs posant encore probleme, il faudra supprimer la valeur de l'attribut de
l'instance ou modifier les restrictions de domaine des classes



[19] L'utilisateur clique sur la classe université.

Consulter la base scolarite

Il reste a regler le probleme pour l(es) attribut(s) ("type-etablis")

La suite est traitée par le mécanisme de révision classique.

c) L'assertion $(o.a=v)$

Cette valeur est inconsistante si elle contredit des restrictions posées sur l'attribut de ses classes de rattachement. Afin d'aider le concepteur à modifier la valeur de l'attribut de l'instance, on lui indique le domaine auquel doit appartenir la valeur, et si ce domaine est fini, les différentes valeurs possibles lui sont proposées.

[19] Les classes `postbac` et `université` sont en vert, et la classe `IUT` en orange.

d) L'assertion ($o.a=o'$)

Il y a inconsistance si l'instance o' ne peut pas appartenir aux classes adéquates, car au moins un de ses valeurs d'attributs ne satisfait pas les contraintes posées sur les classes qui restreignent l'attribut a des classes d'appartenance de o. On indique au concepteur l'ensemble des instances qui appartiennent à l'ensemble des classes qui restreignent l'attribut a de la classe c.

e) L'assertion ($c.a \subseteq d$)

Les problèmes qui se posent peuvent être de nature diverse, chaque type étant résolu de manière spécifique :

- Il y a inconsistance car la valeur d'attribut des instances rattachées à la classe n'appartiennent pas au domaine d. Dans ce cas, on indique l'ensemble des valeurs d'attributs qui sont problématiques.
- La classe c est incohérente si dans le domaine de l'attribut a dans la forme normale avant cet ajout a une intersection vide avec d. Dans ce cas, on indique l'ancien domaine de l'attribut.
- Les sous-classes de c sont incohérentes si le domaine de leur attribut a a une intersection vide avec d. Dans ce cas, on indique les domaines des attributs de toutes les sous-classes qui sont problématiques.

Ces trois cas sont évidemment non disjoints.

Ces informations permettent au concepteur de définir le nouveau domaine selon les connaissances qu'il souhaite préserver sur l'attachement des instances aux classes, les relations de spécialisation ou les domaines d'attribut de classe.

f) L'assertion ($c.a \subseteq c'$)

L'idée est de déterminer la super-classe c'' de c' la plus spécifique, telle que toutes les incohérences et inconsistances soulevées par ($c.a \subseteq c'$) ne soient pas soulevées par ($c.a \subseteq c''$). Il suffit de parcourir les super-classes et de vérifier pour chacune la consistance et la cohérence de l'assertion ($c.a \subseteq c''$)

2. Perdre certaines assertions déduites

Voyons un cas particulier où la recherche de la minimalité de la base ne correspond peut-être pas à ce qui est attendu par le concepteur : si l'assertion ($o.a=o'$) – de laquelle on pouvait déduire précédemment ($o' \in c'$) par la règle de ($\in$-inférence) – est supprimée, pour que la solution soit minimale, il faut que ($o' \in c'$) soit conservée dans les bases révisées. Or, dans ce cas précis, il est peut-être un peu surprenant de préserver une telle déduction vu que cette

inférence n'était faite que pour préserver la consistance et on préférerait plutôt choisir de ne pas conserver cette assertion. C'est le cas où le principe de maintien des justifications (*reason maintenance*) est privilégié par rapport à celui de préservation du maximum de connaissance. Ces deux stratégies sont représentées dans la Figure 64.

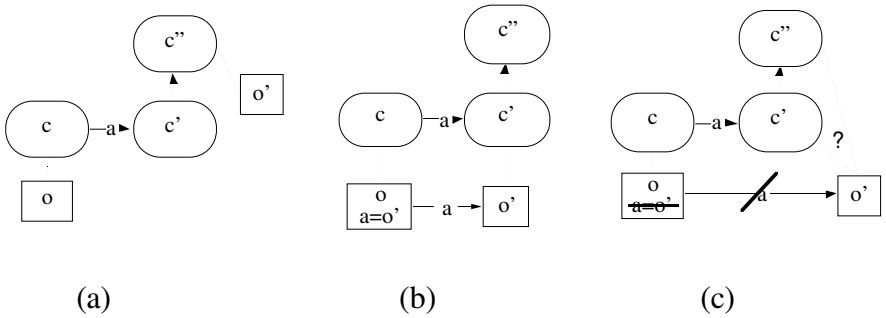


Figure 64 : Si ($o.a=o'$) ∈ Cn(B) (a), la règle de (∈-inférence) permet de déduire ($o' \in c'$) (b) et si lors de la révision ($o.a=o'$) est supprimée doit-on conserver ($o' \in c'$) selon le principe de minimalité, ou la supprimer selon le principe de maintien des justifications (c) ?

Comme il n'est pas possible de déterminer si le concepteur est à la recherche de bases révisées minimales ou s'il souhaite privilégier le principe de maintien des justifications, les deux solutions lui seront proposées. Donc la suppression de valeur d'attribut est :

S(o,a,o')

ou

$B'_- = \{o.a=o'\}$ et $B'_+ = \emptyset$

E. TROPES exprimé dans le langage formel

Le langage qui a été présenté n'est pas celui utilisé par le système de représentation de connaissance TROPES développé au sein du projet Sherpa [Sherpa95] sur lequel la révision a été implémentée. La liaison entre les deux systèmes doit être détaillée et il faut donc considérer en particulier que le système TROPES a pour mission de s'assurer qu'il construit bien une base au sens donné dans le chapitre I.

1. Traduction TROPES – langage à objets

En TROPES, les instances et les valeurs sont partitionnés en un ensemble de concepts et un ensemble de types. Aux valeurs correspond un domaine (qui sera V dans le langage présenté ici) et aux instances un domaine (qui sera O). Les instances des concepts sont munies d'attributs, chaque concept constituant un espace de nom différent pour les noms de ces attributs. Ceux-ci sont transformés en un ensemble d'attributs A dans le langage ci-dessus. Les concepts peuvent être munis de différents points de vue qui sont importants pour nommer

les classes mais n'ont aucune contrepartie dans le langage ci-dessus. Chaque point de vue contient un ensemble de classes ordonnées en un arbre par la relation de spécialisation. L'espace de nommage des classes étant le point de vue, ici encore chaque classe aura un nom unique et l'ensemble des classes constituera donc C. L'ordre correspondant à la spécialisation sera rendu par des assertions de type ($c \leq c'$) et les contraintes posées sur les attributs dans les classes seront traduites par des assertions de type ($c.a \subseteq d$). La racine de chaque hiérarchie recevra donc aussi les contraintes portant sur les attributs du concept. Chaque instance dans O fait partie d'un et d'un seul concept et est rattachée à une unique classe par point de vue ce qui est rendu par des assertions de type ($o \in c$). Il faut noter que l'attachement multiple n'a pas été prohibé jusqu'à maintenant. Enfin, les valeurs d'attributs des instances seront traduites par ($o.a=v$).

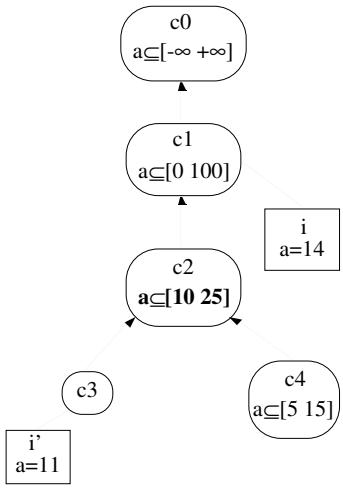


Figure 65 : exemple de base de connaissance (repris de la section II.A)

Voici ce que serait la base de la Figure 65, exprimée dans le formalisme de TROPES.

```
<exemple-figure-1
 bases-requises = {} ;
  types = {"string", "integer"};
  points-de-vue = {};
  concepts = {
    <k
      clefs = {
        <name dans = string; nature = propriete; constructeur = un;>};
      attributs = {
    <a dans = integer; nature = propriete; constructeur = un;>};
```

```
      points-de-vue = {
         <v
            classe-racine =
               <c0 attributs = {<a>};>;
            classes = {
               <c1 super-classe = c0;
                  attributs = {<a intervalles = {[0 100]};>};>,
               <c2 super-classe = c1;
                  attributs = {<a intervalles = {[10 25]};>};>,
               <c3 super-classe = c2;
                  attributs = {<a>};>,
               <c4 super-classe = c2;
                  attributs = {<a intervalles = {[5 15]};>};>};>};
      passerelles = {};>};
   fichiers-instances = {
      <k name = "i"; est-dans = {c2@v}; a = 14;>,
      <k name = "i'"; est-dans = {c3@v}; a = 11;>,
      <k name = "i""; est-dans = {c4@v};>} ;
   modules-talk = {} ;
```

TROPES est capable de vérifier un certain nombre de propriétés dans la construction des bases : que les noms sont uniques, que les instances n'appartiennent qu'à un seul concept, que les valeurs et domaines des attributs sont bien typés, que les taxonomies sont des arbres, etc. Il est capable d'émettre des messages d'erreurs si l'une des ses propriétés n'est pas satisfaite.

Une fois ce premier niveau de vérification passé, la base peut être considérée comme une base au sens énoncé ci-dessus et les mécanismes de révision pourront s'appliquer en retranchant ou ajoutant des assertions à la base TROPES. Le langage d'expression accepte aussi des bases qui ne sont pas des bases TROPES (par exemple, la spécialisation multiple n'a pas été prohibée) mais il suffit, pour détecter des inconsistances et proposer des révisions dans les bases TROPES, de pouvoir exprimer celles-ci dans le langage proposé et de ne pas tenir compte des bases révisées qui ne sont pas des bases TROPES.

2. Principales différences

La multi-spécialisation est autorisée, alors que ce n'est pas le cas dans TROPES. Mais l'algorithme décrit précédemment tient compte de cette particularité de TROPES.

TROPES permet de manipuler divers types de données pas forcément aussi simples que les entiers (des types tels que les ensembles de dates). Par ailleurs, le langage d'expression des types est plus riche que les intervalles d'entiers (par exemple, TROPES autorise l'expression des unions d'intervalles pour les entiers).

Tenir compte des concepts est relativement aisé vu que plusieurs hiérarchies peuvent coexister dans le langage d'objets défini; il suffit de définir une hiérarchie par concept. La vérification qu'une instance appartient à un concept et un seul devra cependant être faite.

Les points de vue peuvent s'exprimer de la même manière. Quant aux passerelles, qui sont une extension plus typique de TROPES, elles peuvent correspondre à un nouveau type d'assertion ($c_1 \cap \ldots \cap c_n \subseteq c$) qui sera interprétée comme $I[c_1] \cap \ldots \cap I[c_n] \subseteq I[c]$. Le système déductif ne nécessite aucune modification particulière, car l'assertion caractérisant une passerelle ne sera satisfaite que dans le cas où elle est effectivement donnée dans la base (on pourra toujours trouver une interprétation ne satisfaisant pas l'assertion si elle n'appartient pas à la base). L'ajout de ce nouveau type d'assertion semble donc avoir pour seule conséquence la diminution du nombre d'interprétations valides et donc du nombre de conséquences, mais sans effet sur la formalisation proposée (par contre, l'effet va se ressentir dans la définition de l'inconsistance et dans les stratégies de révision mises en œuvre).

3. Révision dans TROPES : des adaptations

L'hypothèse de mono-spécialisation est posée dans TROPES, et les algorithmes ont été adaptés à cette hypothèse. Une base avec deux relations de spécialisation pour la même classe (dans la forme première) n'est donc pas acceptable et lors de la révision, l'une des deux assertions de relation de spécialisation devra être supprimée même si elle préserve la consistance de la base. La notion de minimalité semble donc modifiée.

Cependant, afin de rendre compte de ces nouvelles contraintes, il suffit de restreindre la notion de base acceptable aux bases ne possédant pas de classe ayant deux super-classes ni d'instances appartenant à deux classes du même point de vue. Dans ce cas, la définition de la minimalité est toujours valable.

F. Exemples de révision

Les exemples qui suivent montrent la révision d'une base de connaissance suite à l'ajout d'une assertion du type (*o.a=o'*) qui est inconsistante; c'est l'exemple le plus compliqué car les modifications d'instances, aussi bien que les modifications de classes permettent de résoudre l'inconsistance. Nous donnons la base initiale et plusieurs exécutions significatives de l'algorithme.

EXEMPLE 1

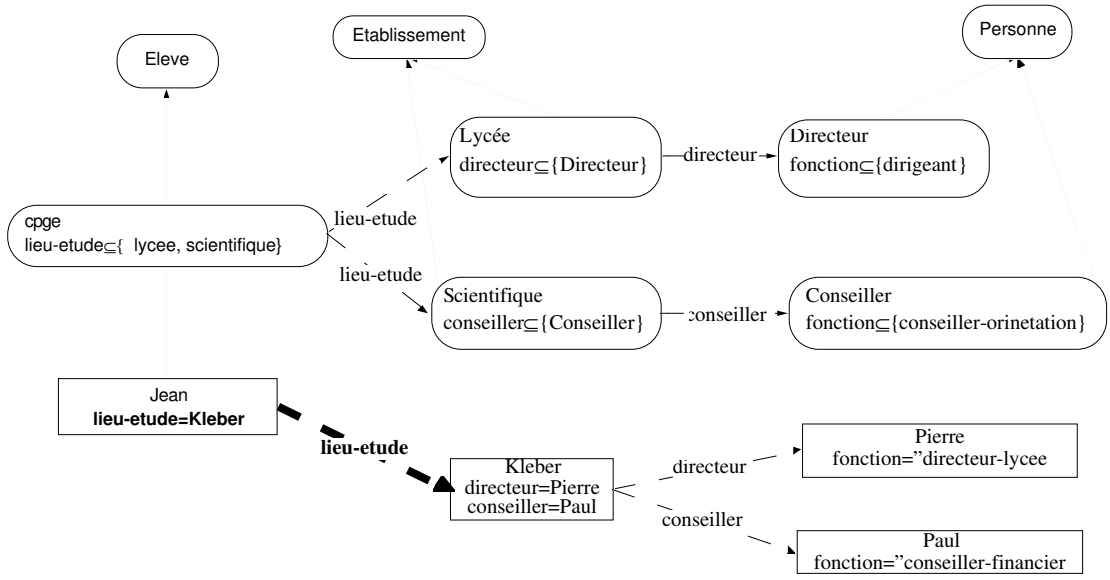
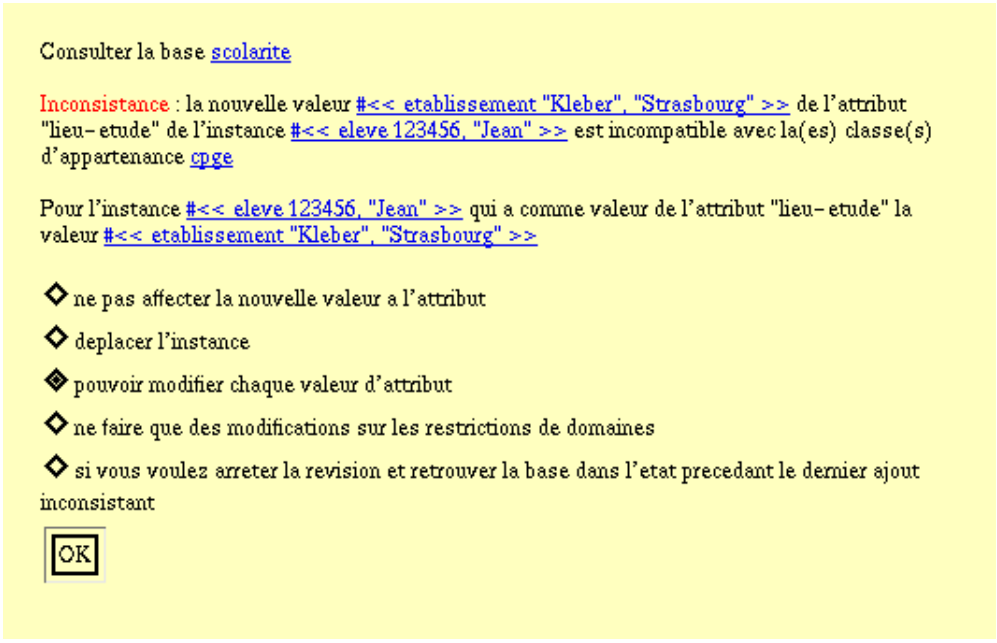
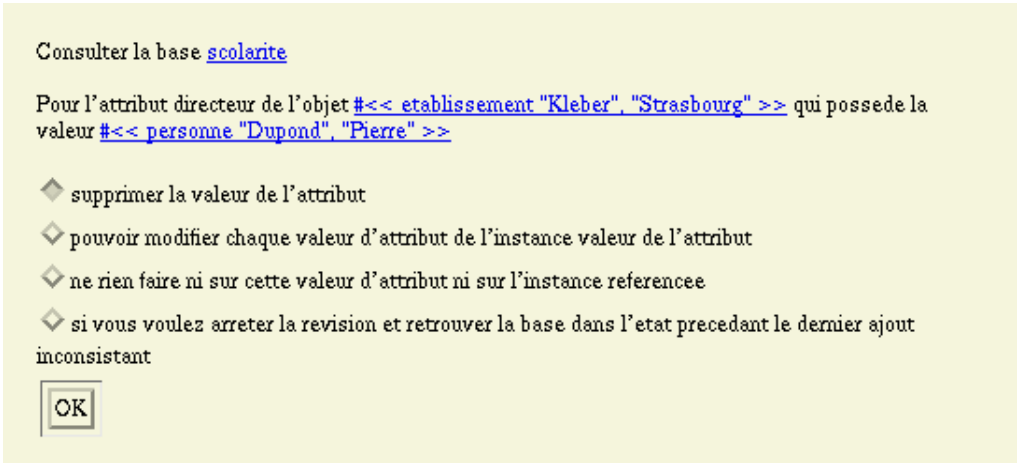
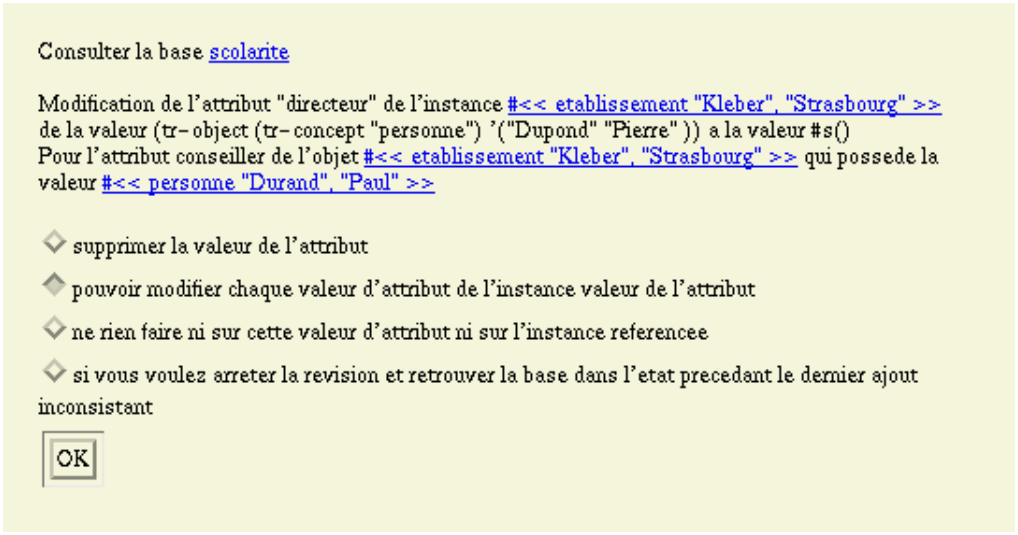


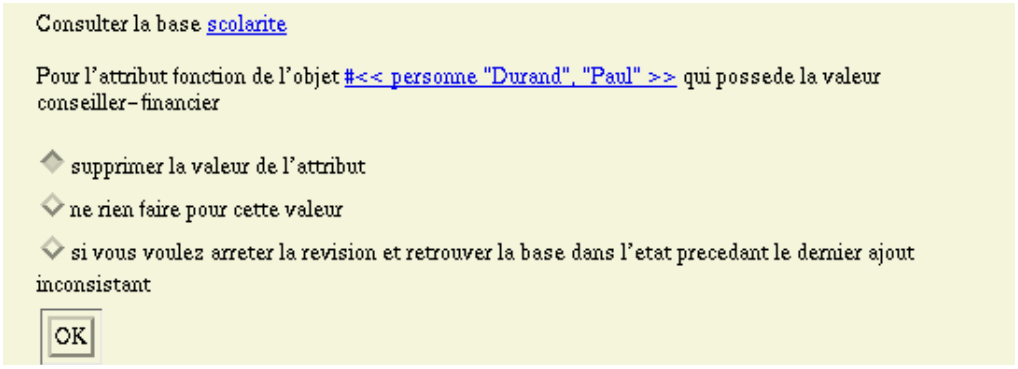
Figure 66 : Exemple de base de connaissance suite à l'ajout de Jean.lieu-etude=Kleber. Deux inconsistances sont déclenchées :
Jean.lieu-etude.directeur.fonction="directeur-lycee" et cpge.lieu-etude.directeur.fonction⊆{dirigeant},
Jean.lieu-etude.conseil.fonction="conseiler-financier" et cpge.lieu-etude.conseil.fonction⊆{cons-orientation}

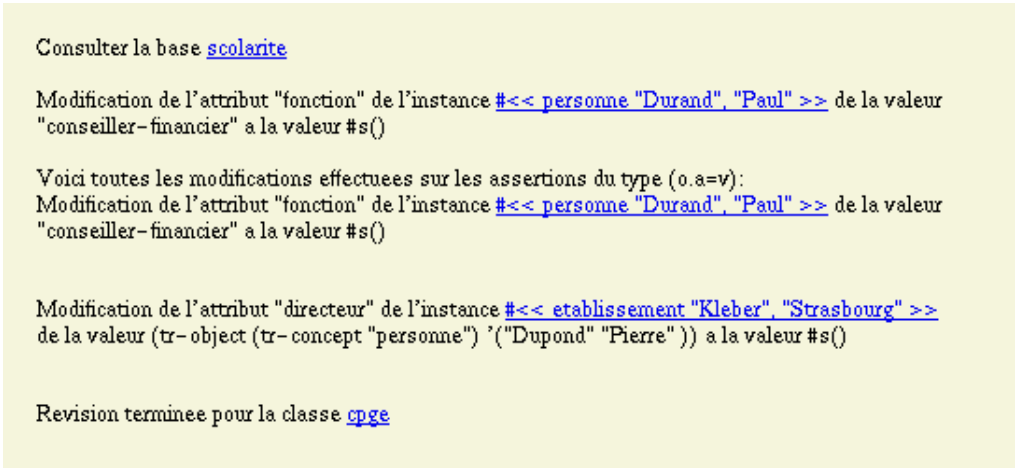
EXEMPLE 1.1


Consulter la base scolarite
Inconsistance : la nouvelle valeur #<< etablissement "Kleber", "Strasbourg" >> de l'attribut "lieu-etude" de l'instance #<< eleve 123456, "Jean" >> est incompatible avec la(es) classe(s) d'appartenance cpge
Pour l'instance #<< eleve 123456, "Jean" >> qui a comme valeur de l'attribut "lieu-etude" la valeur #<< etablissement "Kleber", "Strasbourg" >>
ne pas affecter la nouvelle valeur a l'attribut
deplacer l'instance
pouvoir modifier chaque valeur d'attribut
ne faire que des modifications sur les restrictions de domaines
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
Pour l'attribut directeur de l'objet #<< etablissement "Kleber", "Strasbourg" >> qui possede la valeur #<< personne "Dupond", "Pierre" >>
supprimer la valeur de l'attribut
pouvoir modifier chaque valeur d'attribut de l'instance valeur de l'attribut
ne rien faire ni sur cette valeur d'attribut ni sur l'instance referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK







Le résultat de cette révision (suppressions de Paul.fonction="conseiller-financier" et de Kleber.directeur=Pierre) est illustrée dans la Figure 67.

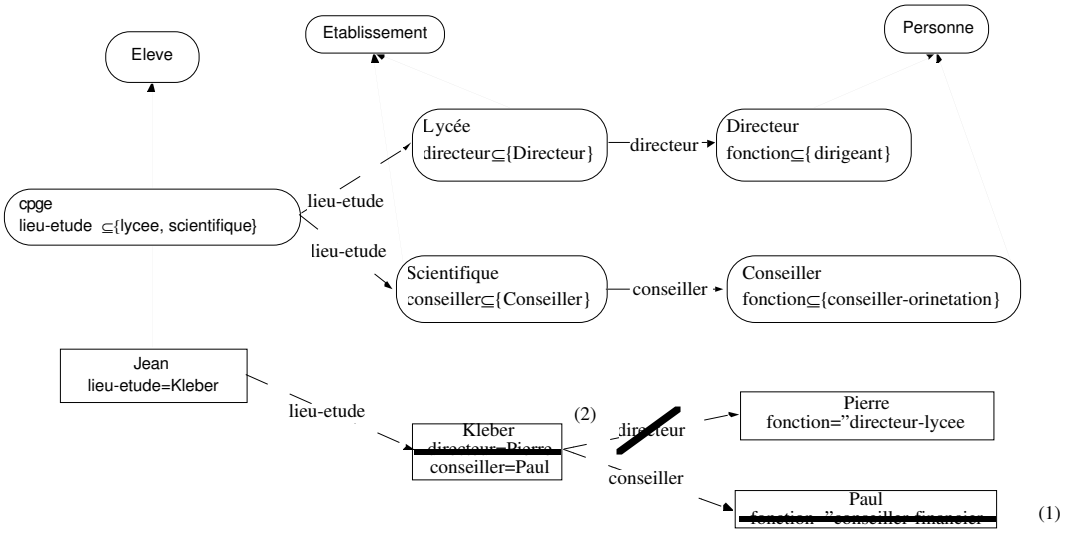


Figure 67 : Exemple de révision de la base de la Figure 66 par suppression de Kleber.directeur=Pierre et Paul.fonction="conseiller-financier"

EXEMPLE 1.2

Voyons une deuxième exécution de la révision sur le même exemple, en modifiant les classes cette fois-ci. Il est bien entendu possible de combiner les solutions obtenues par ces deux exécutions.

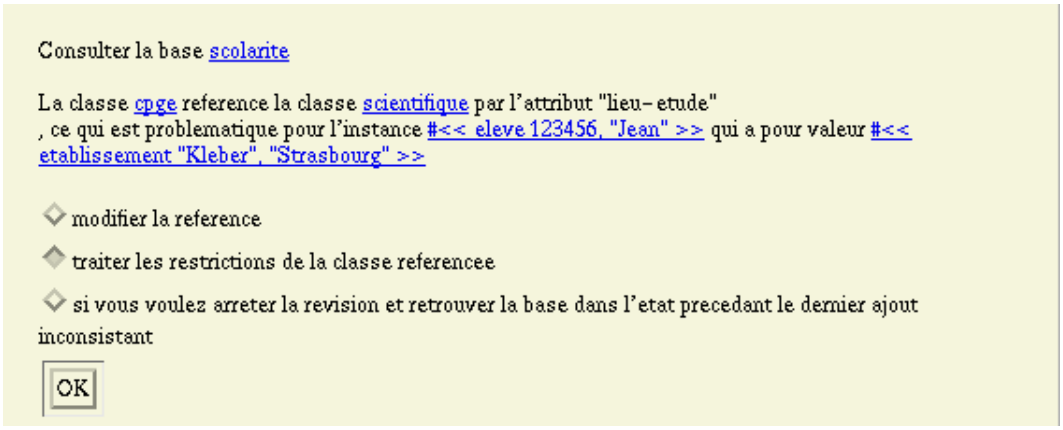
Consulter la base scolarite

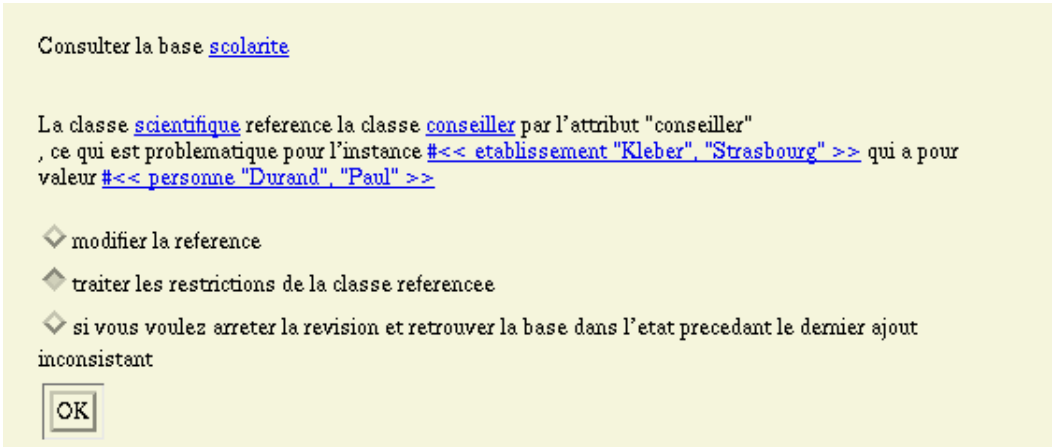
Inconsistance : la nouvelle valeur #<< etablissement "Kleber", "Strasbourg" >> de l'attribut "lieu-etude" de l'instance #<< eleve 123456, "Jean" >> est incompatible avec la(es) classe(s) d'appartenance cpge

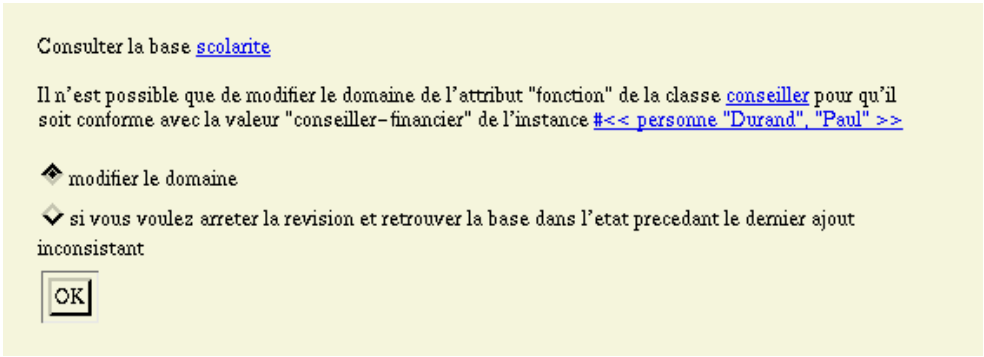
Pour l'instance #<< eleve 123456, "Jean" >> qui a comme valeur de l'attribut "lieu-etude" la valeur #<< etablissement "Kleber", "Strasbourg" >>

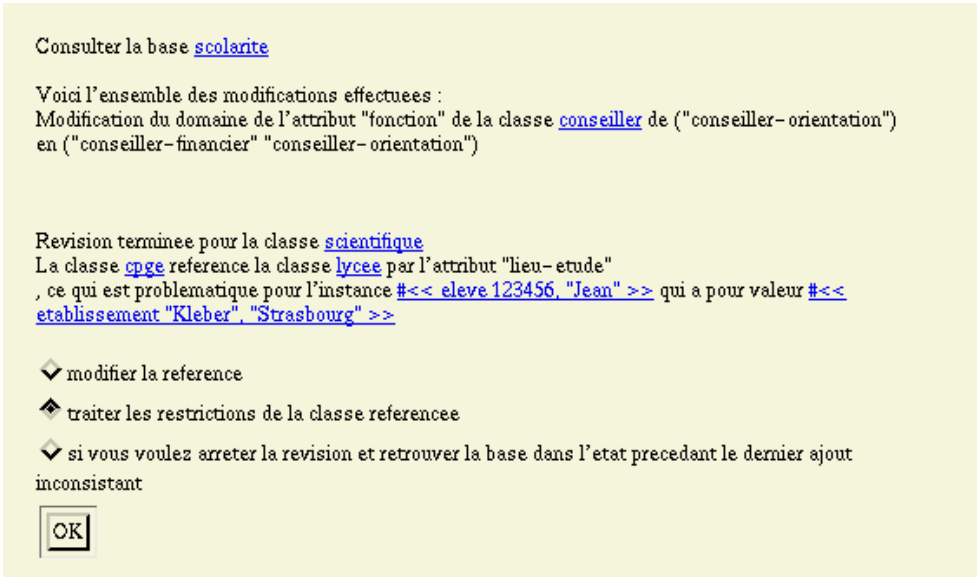
- ◇ ne pas affecter la nouvelle valeur a l'attribut
- ◇ deplacer l'instance
- ◇ pouvoir modifier chaque valeur d'attribut
- ◈ ne faire que des modifications sur les restrictions de domaines
- ◇ si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant

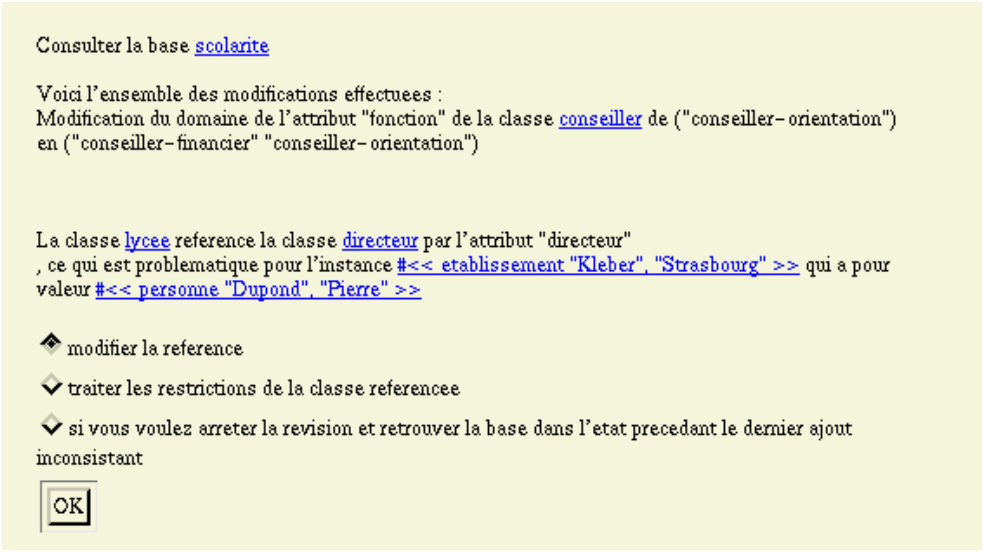
OK


Consulter la base scolarite
La classe cpge reference la classe scientifique par l'attribut "lieu-etude"
, ce qui est problematique pour l'instance #<< eleve 123456, "Jean" >> qui a pour valeur #<< etablissement "Kleber", "Strasbourg" >>
modifier la reference
traiter les restrictions de la classe referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
La classe scientifique reference la classe conseiller par l'attribut "conseiller"
, ce qui est problematique pour l'instance #<< etablissement "Kleber", "Strasbourg" >> qui a pour valeur #<< personne "Durand", "Paul" >>
modifier la reference
traiter les restrictions de la classe referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
Il n'est possible que de modifier le domaine de l'attribut "fonction" de la classe conseiller pour qu'il soit conforme avec la valeur "conseiller-financier" de l'instance #<< personne "Durand", "Paul" >>
modifier le domaine
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
Voici l'ensemble des modifications effectuees :
Modification du domaine de l'attribut "fonction" de la classe conseiller de ("conseiller-orientation") en ("conseiller-financier" "conseiller-orientation")
Revision terminee pour la classe scientifique
La classe cpge reference la classe lycee par l'attribut "lieu-etude"
, ce qui est problematique pour l'instance #<< eleve 123456, "Jean" >> qui a pour valeur #<< etablissement "Kleber", "Strasbourg" >>
modifier la reference
traiter les restrictions de la classe referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
Voici l'ensemble des modifications effectuees :
Modification du domaine de l'attribut "fonction" de la classe conseiller de ("conseiller-orientation") en ("conseiller-financier" "conseiller-orientation")
La classe lycee reference la classe directeur par l'attribut "directeur"
, ce qui est problematique pour l'instance #<< etablissement "Kleber", "Strasbourg" >> qui a pour valeur #<< personne "Dupond", "Pierre" >>
modifier la reference
traiter les restrictions de la classe referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK

Consulter la base scolarite

Voici l'ensemble des modifications effectuees :
Modification du type de l'attribut "directeur" de la classe lycee de la(des) classe(s) directeur vers la(es) classe(s) personne

Modification du domaine de l'attribut "fonction" de la classe conseiller de ("conseiller-orientation") en ("conseiller-financier" "conseiller-orientation")

Revision terminee pour la classe cpge

La revision est terminee

Le résultat de cette exécution est illustré sur la Figure 68.

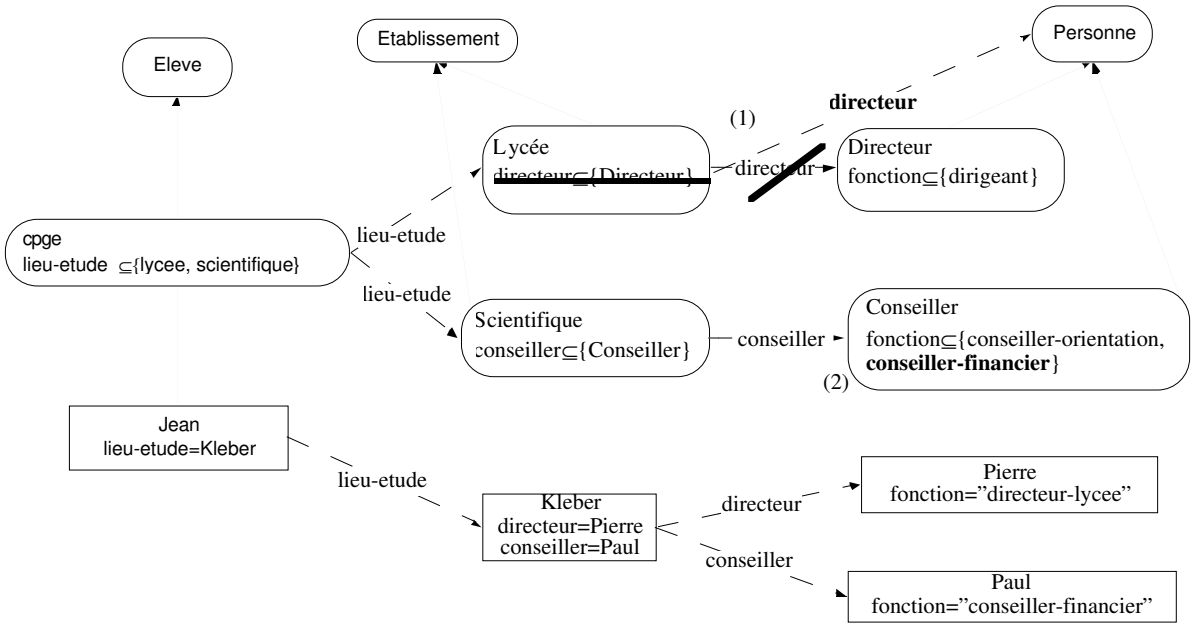


Figure 68 : exemple de révision de la base illustrée par la Figure 66; lycée.directeur⊆Directeur a été modifié en lycée.directeur⊆Personne et Conseiller.fonction⊆{conseiller-orientation} a été modifié en Conseiller.fonction⊆{conseiller-financier, conseiller-orientation}.

EXEMPLE 2

Voyons un autre exemple qui met en évidence la répercussion d'une suppression sur l'ensemble des autres inconsistances non encore résolues (Figure 69).

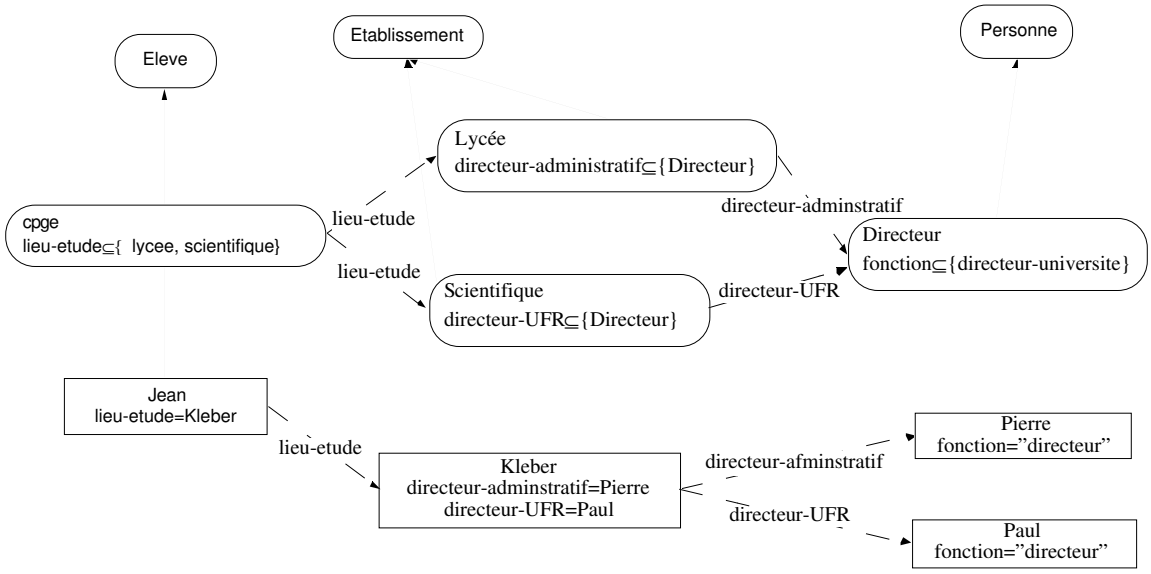
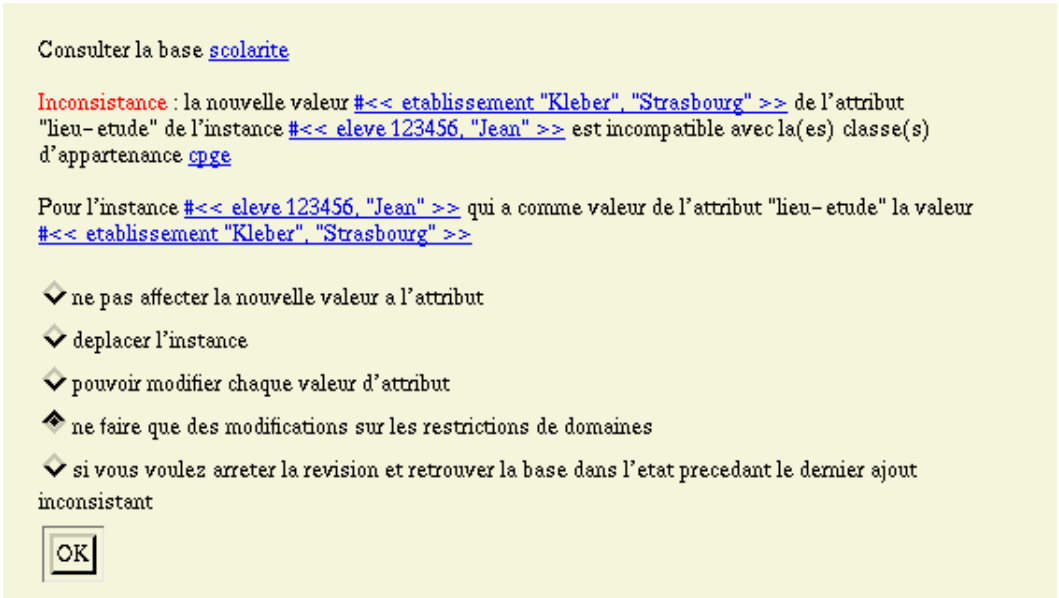
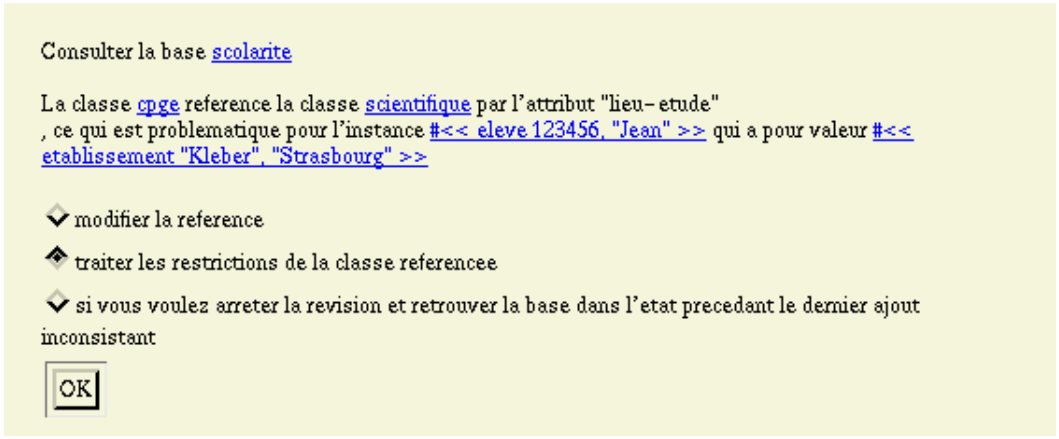


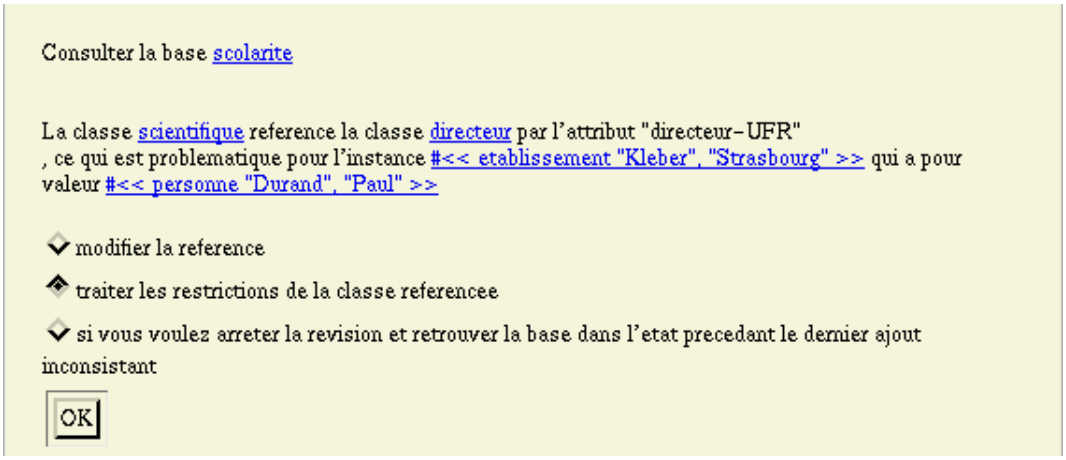
Figure 69 : exemple de base dans lequel l'ajout de Jean.lieu-etude=Kleber soulève les inconsistances suivantes : Jean.lieu-etude.directeur-adminstratif.fonction="directeur" et cpge.lieu-etude.directeur-adminstratif.fonction⊆{directeur-universite}; Jean.lieu-etude.directeur-UFR.fonction="directeur" et cpge.lieu-etude.directeur-UFR.fonction⊆"directeur-universite"

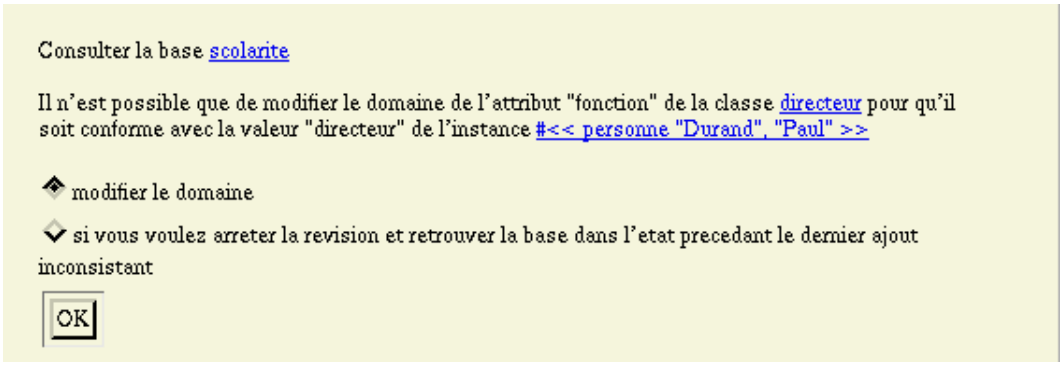
EXEMPLE 2.1

Le premier exemple (Figure 70) de révision montre que la suppression d'un domaine est propagé à l'ensemble des inconsistances pour vérifier si elles ne sont pas éventuellement résolues.




Consulter la base scolarite
La classe cpge reference la classe scientifique par l'attribut "lieu-etude"
, ce qui est problematique pour l'instance #<< eleve 123456, "Jean" >> qui a pour valeur #<< etablissement "Kleber", "Strasbourg" >>
modifier la reference
traiter les restrictions de la classe referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
La classe scientifique reference la classe directeur par l'attribut "directeur-UFR"
, ce qui est problematique pour l'instance #<< etablissement "Kleber", "Strasbourg" >> qui a pour valeur #<< personne "Durand", "Paul" >>
modifier la reference
traiter les restrictions de la classe referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
Il n'est possible que de modifier le domaine de l'attribut "fonction" de la classe directeur pour qu'il soit conforme avec la valeur "directeur" de l'instance #<< personne "Durand", "Paul" >>
modifier le domaine
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK

```
Consulter la base scolarite

Voici l'ensemble des modifications effectuees :
Modification du domaine de l'attribut "fonction" de la classe directeur de ("directeur-ecole"
"directeur-universite") en ("directeur" "directeur-ecole" "directeur-universite")


Revision terminee pour la classe scientifique
Revision terminee pour la classe cpge

La revision est terminee
```

Le résultat de cette exécution est illustré par la Figure 70.

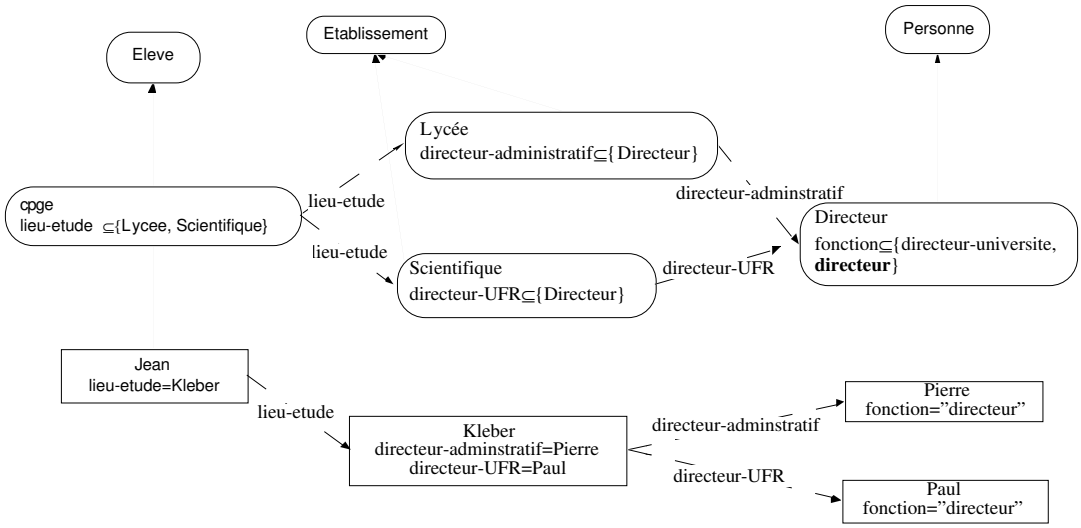
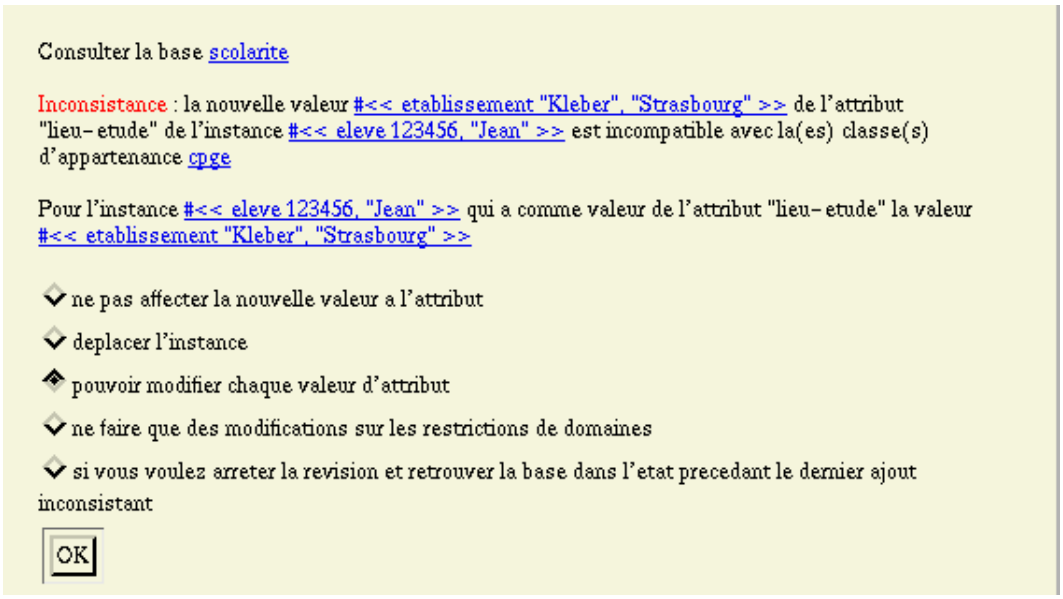
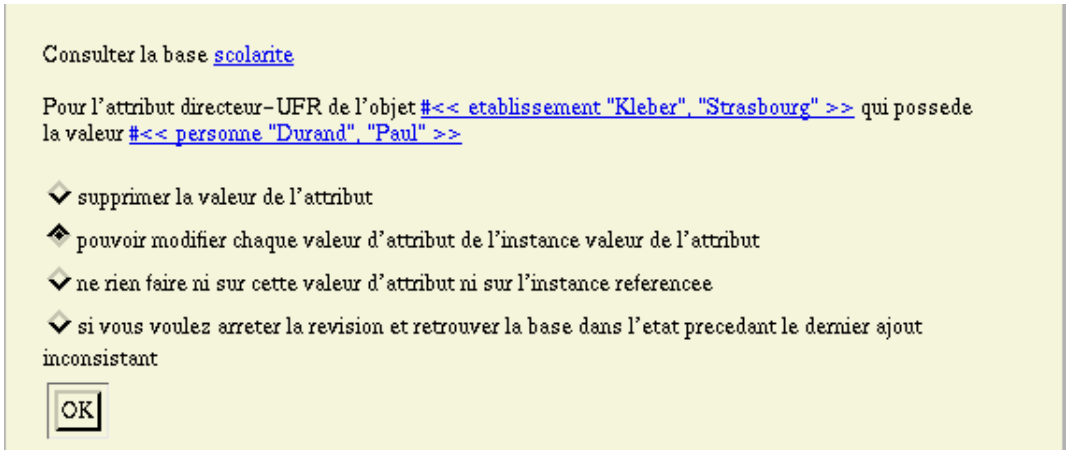


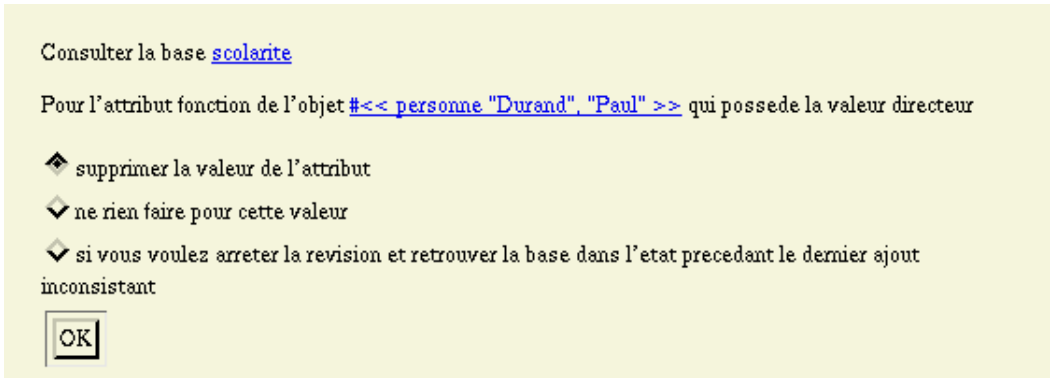
Figure 70 : La résolution de l'inconsistance portant sur Pierre, on modifie la restriction de l'attribut fonction de la classe Directeur et ceci résout également l'inconsistance sur l'instance Paul.

EXEMPLE 2.2

Le second exemple de révision (Figure 71) montre que lors d'une suppression de restriction d'attribut d'une classe, l'algorithme vérifie que les suppressions précédentes d'assertions du type *o.a=o'* ne sont pas superflues.


Consulter la base scolarite
Inconsistance : la nouvelle valeur #<< etablissement "Kleber", "Strasbourg" >> de l'attribut "lieu-etude" de l'instance #<< eleve 123456, "Jean" >> est incompatible avec la(es) classe(s) d'appartenance cpge
Pour l'instance #<< eleve 123456, "Jean" >> qui a comme valeur de l'attribut "lieu-etude" la valeur #<< etablissement "Kleber", "Strasbourg" >>
ne pas affecter la nouvelle valeur a l'attribut
deplacer l'instance
pouvoir modifier chaque valeur d'attribut
ne faire que des modifications sur les restrictions de domaines
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
Pour l'attribut directeur-UFR de l'objet #<< etablissement "Kleber", "Strasbourg" >> qui possede la valeur #<< personne "Durand", "Paul" >>
supprimer la valeur de l'attribut
pouvoir modifier chaque valeur d'attribut de l'instance valeur de l'attribut
ne rien faire ni sur cette valeur d'attribut ni sur l'instance referencee
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK


Consulter la base scolarite
Pour l'attribut fonction de l'objet #<< personne "Durand", "Paul" >> qui possede la valeur directeur
supprimer la valeur de l'attribut
ne rien faire pour cette valeur
si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant
OK

Consulter la base scolarite

Modification de l'attribut "fonction" de l'instance #<< personne "Durand", "Paul" >> de la valeur "directeur" a la valeur #s()
Pour l'attribut directeur-adm de l'objet #<< etablissement "Kleber", "Strasbourg" >> qui possede la valeur #<< personne "Dupond", "Pierre" >>

- supprimer la valeur de l'attribut
- pouvoir modifier chaque valeur d'attribut de l'instance valeur de l'attribut
- ne rien faire ni sur cette valeur d'attribut ni sur l'instance referencee
- si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant

OK

Consulter la base scolarite

Voici toutes les modifications effectuees sur les assertions du type (o.a=v):
Voici l'ensemble des modifications effectuees :
Modification de l'attribut "fonction" de l'instance #<< personne "Durand", "Paul" >> de la valeur "directeur" a la valeur #s()

il faut a present modifier les domaines pour lever les inconsistances qui subsistent
La classe cpge reference la classe lycee par l'attribut "lieu-etude"
, ce qui est problematique pour l'instance #<< eleve 123456, "Jean" >> qui a pour valeur #<< etablissement "Kleber", "Strasbourg" >>

- modifier la reference
- traiter les restrictions de la classe referencee
- si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant

OK

Consulter la base scolarite

La classe lycee reference la classe directeur par l'attribut "directeur-adm"
, ce qui est problematique pour l'instance #<< etablissement "Kleber", "Strasbourg" >> qui a pour valeur #<< personne "Dupond", "Pierre" >>

- modifier la reference
- traiter les restrictions de la classe referencee
- si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant

OK

Consulter la base scolarite

Il n'est possible que de modifier le domaine de l'attribut "fonction" de la classe directeur pour qu'il soit conforme avec la valeur "directeur" de l'instance #<< personne "Dupond", "Pierre" >>

◆ modifier le domaine

◆ si vous voulez arreter la revision et retrouver la base dans l'etat precedant le dernier ajout inconsistant

OK

Consulter la base scolarite

Voici l'ensemble des modifications effectuees :
Modification de l'attribut "fonction" de l'instance #<< personne "Durand", "Paul" >> de la valeur #s() a la valeur "directeur"

Modification du domaine de l'attribut "fonction" de la classe directeur de ("directeur-ecole" "directeur-universite") en ("directeur" "directeur-ecole" "directeur-universite")

Modification de l'attribut "fonction" de l'instance #<< personne "Durand", "Paul" >> de la valeur "directeur" a la valeur #s()

Revision terminee pour la classe lycee
Revision terminee pour la classe cpge

La revision est terminee

Il est à noter que la deuxième partie de la révision résout également la première inconsistance et donc que la valeur d'attribut de l'instance Paul est rétablie.

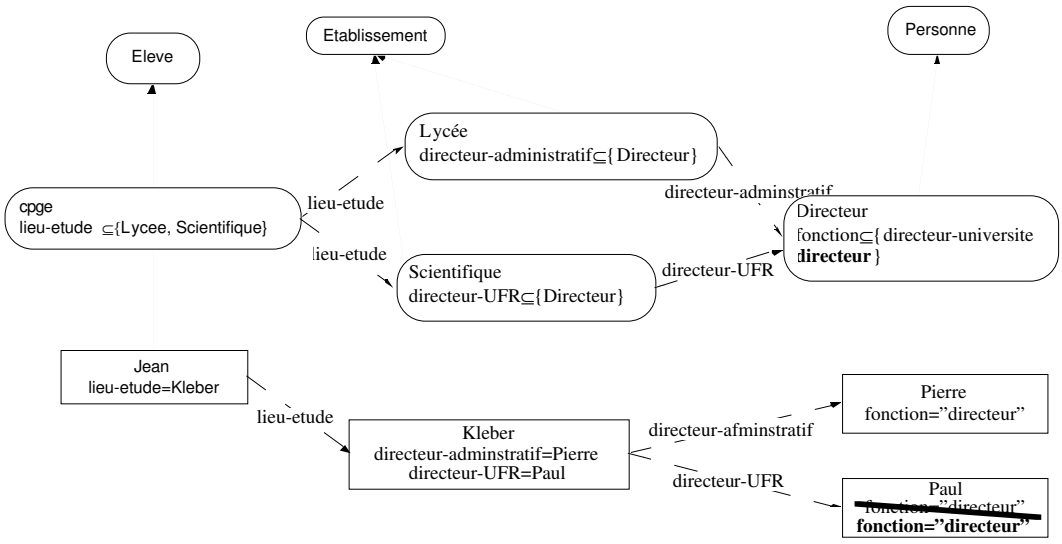


Figure 71 : exemple de résolution où l'on supprime Paul.fonction="directeur", puis lors de la révision pour l'instance Pierre, la restriction d'attribut de domaine de Directeur est modifié, ce qui permet de rétablir Paul.fonction="directeur".

G. Des fonctionnalités supplémentaires

Pour améliorer l'utilisation du module de révision, l'utilisateur a la possibilité de défaire les pas de modification (paragraphe 1) et de visualiser toutes les modifications effectuées sur la base lors du processus de révision (paragraphe 2).

1. Undo

Comme déjà expliqué, l'ensemble des bases révisées n'est pas fourni au concepteur, mais il lui est proposé de réviser la base en différentes étapes (par exemple, d'abord choisir un déplacement de classes avant de s'occuper des instances qui sont encore problématiques). Le processus de révision se déroule alors par choix dans des écrans successifs, ce qui fait que le concepteur ne peut pas forcément mesurer l'étendue des modifications nécessaires à la révision de la base suite à un de ces choix. Ceci n'est bien évidemment pas satisfaisant d'un point de vue ergonomique, mais n'a pas été résolu. Cependant, afin de limiter ce désagrément, nous avons mis en place un mécanisme de *undo*, afin de pouvoir revenir sur les points de choix, aussi loin que possible.

D'un point de vue algorithmique, les actions de modifications de la base sont mises dans une pile et à chaque type d'action est associée l'action réciproque (c'est-à-dire celle qui permet de revenir à l'état antérieur de la base). Lors d'un retour arrière, les actions sont dépilées les unes après les autres en effectuant sur la base leur action réciproque associée.

2. Affichage des modifications

La pile créée pour réaliser le retour-arrière permet également l'affichage de toutes les modifications réalisées sur une base après sa révision complète.

Ainsi, l'ensemble des modifications effectuées par la révision de la base dans l'exemple 2.2 est :

```
Voici l'ensemble des modifications effectuees :
Modification de l'attribut "fonction" de l'instance #<< personne "Durand", "Paul" >> de la valeur
#s() a la valeur "directeur"


Modification du domaine de l'attribut "fonction" de la classe directeur de ("directeur-ecole"
"directeur-universite") en ("directeur" "directeur-ecole" "directeur-universite")


Modification de l'attribut "fonction" de l'instance #<< personne "Durand", "Paul" >> de la valeur
"directeur" a la valeur #s()
```

Conclusion

L'objectif de ce travail était de définir un système de révision pour les représentations de connaissance par objets qui puisse être implémenté. Deux problèmes devaient alors être résolus :

- Définir un système de révision spécifique à une représentation de connaissance par objets.
- Obtenir un programme utilisable en pratique alors que, dans le cadre logique, la trop grande complexité fait renoncer à toute implémentation.

Nous avons résolu le premier problème en définissant, tout d'abord, un langage d'objets, pour lequel nous avons donné la syntaxe, la sémantique, ainsi qu'un système déductif correct et complet pour les bases consistantes. De plus, nous avons défini la forme première d'une base de connaissance : forme normalisée unique, finie, qui permet de déduire l'ensemble de la base et dans laquelle les suppressions d'assertion sont effectives.

Dans un second temps, nous avons donné une caractérisation syntaxique des problèmes pouvant survenir dans une base, à savoir des inconsistances – ensemble d'assertions pour lesquelles il n'existe pas de modèle de la base – et des incohérences – des classes auxquelles aucune instance ne peut être attachée. La forme première a alors permis de décrire les bases révisées d'une base suite à un ajout d'assertion inacceptable, c'est-à-dire inconsistante ou incohérente. La révision se décompose en l'obtention d'une base contractée, suivie de l'ajout de la dernière assertion. La contraction comporte deux étapes : supprimer un ensemble d'assertions de la forme première pour obtenir un ensemble consistant, puis ajouter les assertions qui appartenaient à la clôture déductive de la base initiale mais qui ne sont plus déductibles dans la base contractée tout en y étant acceptables. Les bases révisées sont telles qu'elles préservent le maximum des assertions de la clôture déductive de la base initiale, au sens de l'inclusion ensembliste. Cette notion de minimalité a été décrite de manière syntaxique et sémantique. De plus, nous en avons donné une interprétation qui se fonde sur la structure de la base : ainsi, par exemple une classe est toujours déplacée sous sa super-classe la plus spécifique pouvant l'accueillir ; de même, les modifications des restrictions d'attributs des classes sont également les plus limitées possibles, le nouveau domaine restreignant l'attribut étant le plus proche possible, au sens de l'inclusion ensembliste, de l'ancien.

Cette définition de révision satisfait les principaux postulats énoncés dans le cadre logique, à savoir : les bases révisées sont consistantes et elles contiennent la dernière assertion ; de plus elles satisfont un critère de minimalité, même si celui-ci ne permet pas la détermination d'une base révisée unique. La révision définie correspond à l'opérateur de révision défini à partir de la contraction maxi-choix [Alchourròn&85], à la différence près que le choix parmi les différentes solutions possibles n'est pas arbitraire mais laissé au concepteur de la base. Le choix du concepteur est plus facile que dans un langage logique, car il est guidé par la structure de la base, le type d'assertions qu'il veut conserver et les informations telles que la liste des inconsistances transmises par le processus de révision.

Le second problème, concernant l'utilisation concrète de l'algorithme, a trouvé une solution en différents points :

- L'interactivité avec le concepteur de la base est sans doute le point crucial. Elle permet d'obtenir une base révisée en une complexité polynomiale. Le concepteur de la base obtient, par des choix successifs entre différentes modifications, la base révisée conservant les assertions qu'il ne souhaite pas supprimer.
- Inconsistance et incohérence sont caractérisées syntaxiquement, ce qui permet de les détecter de manière efficace.
- L'algorithme est extrêmement guidé par la structure de la base – les liens de spécialisation entre les classes, les liens de référence entre classes ou instances, le rattachement des instances aux classes –, ce qui permet de ne pas explorer toute la base pour déterminer les assertions à supprimer pour obtenir les bases révisées.
- Enfin, la possibilité de rejeter ou de modifier l'assertion problématique et d'obtenir des bases révisées qui ne satisfont pas le principe de minimalité pour certaines règles d'inférence, permet d'enrichir le système de révision.

Le test de l'implémentation sur des cas vraisemblables montre que les programmes développés peuvent être utilisés de manière concrète. Ceci peut être contrasté avec les travaux en révision logique qui s'achèvent trop souvent sur des résultats décourageants de complexité dans le pire cas et qui n'aboutissent pas à des réalisations manipulables.

Perspectives

Ce travail ouvre différentes perspectives de recherche.

Notion de préférence

Grâce au processus de révision décrit, un ensemble de bases révisées minimales est proposé au concepteur. La minimalité définie ne permet pas d'obtenir une seule base en résultat, contrairement à la révision en logique. Pour proposer une base unique la plus satisfaisante selon des critères de préférence, il faut que ces critères soient disponibles pour le processus de révision, ce qui requiert un important travail. Il faut donc trouver un compromis entre la quantité de connaissances demandées au concepteur et le nombre de bases révisées proposées. L'idée est que le concepteur fournisse des critères supplémentaires afin de restreindre le nombre de bases proposées (Figure 72).

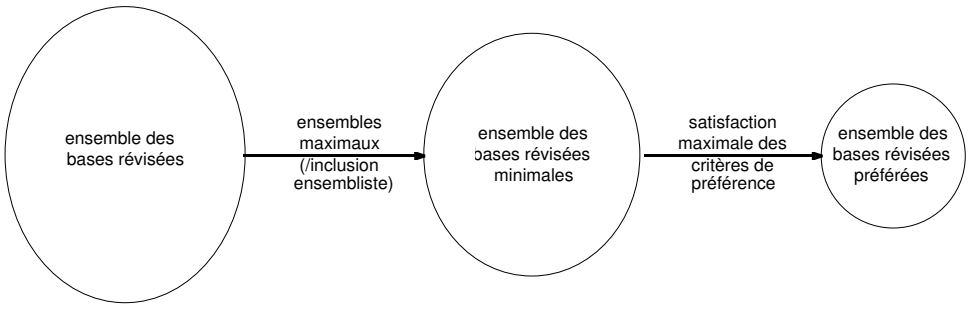


Figure 72 : Restriction du nombre de bases révisées, tout d'abord par le critère de minimalité, puis par les critères de préférence.

Le classement de toutes les informations contenues dans la base selon l'importance que les concepteurs leur accordent est lié à l'ordre selon la pertinence (*relevance ordering*) décrit par [Nebel94]. Cela est cependant fastidieux et demande une connaissance de la base telle que le mécanisme de révision perd de son utilité; c'est pourquoi il est envisagé de se servir des différents types d'assertions.

Connaissance structurelle / connaissance individuelle : Dans le cadre des RCO, il existe deux grandes classes de connaissance: la connaissance structurelle liée à la définition de classes et aux liens établis entre elles ($c \leq c'$, $c.a \subseteq d$ ou $c.a \subseteq c'$) et la connaissance individuelle liée aux instances ($o \in c$, $o.a = o'$ ou $o.a = v$). Selon le contexte d'utilisation de la base (construction de la base à partir d'exemples ou utilisation d'une base dont la structure a été éprouvée), l'un ou l'autre de ces types de connaissance peut être le plus fiable et donc à modifier le moins possible. Si un tel ordre entre la

connaissance structurelle et individuelle a été établi, le choix des bases révisées doit tenir compte de ce critère, en préservant les assertions décrites comme les plus fiables.

Points de vue : TROPES admet des points de vue dans sa représentation (décrits au paragraphe I.C.2). Il est envisageable pour un concepteur de ne vouloir – ou n'être autorisé à – modifier que les informations contenues dans un point de vue donné. Lors de la révision de la base, les solutions proposées ne devront donc concerner que ces informations. Une telle révision selon les points de vue est proche des travaux sur la fusion d'informations provenant de différentes sources, la fiabilité de ces sources dépendant des thèmes sur lesquels elles sont compétentes [Cholvy94].

Extension des corrections proposées

En plus de la révision et des modifications de l'assertion ajoutée, on peut envisager de créer de nouvelles classes, ce qui n'est pas possible dans la révision, car l'un des postulats (*confinement*) stipule qu'aucune nouvelle connaissance ne doit être ajoutée. Ceci semble cependant utile pour les concepteurs. Ces nouvelles classes peuvent être en particulier des copies de classes qui sont modifiées. [Foo95] propose également de raffiner des classes existantes, c'est-à-dire de construire de nouvelles sous-classes, ou de décomposer des classes, c'est-à-dire de remplacer une classe par un ensemble de classes.

Application aux langages de programmation, à la mise à jour

Il serait intéressant d'étudier l'application de ce travail aux langages de programmation objets, afin de pouvoir proposer une aide aux programmeurs. D'autre part, l'application à la mise à jour pourrait permettre d'envisager la modification d'une base de connaissance pour suivre l'évolution du domaine représenté.

Extension du langage de représentation

Le langage considéré rassemble déjà une bonne partie des caractéristiques des représentations de connaissance par objets. Cependant, certaines, que l'on peut en particulier trouver dans TROPES, nécessitent une extension du langage. Il faut pouvoir considérer divers types de données pas forcément aussi simples à manipuler que les entiers; TROPES autorise des types tels que les dates. Cette extension est relativement aisée car le module gérant les types met à disposition les fonctions nécessaires, comme l'intersection de deux types, le test d'appartenance d'une valeur à un domaine, …

Une extension plus typique de TROPES concerne l'ajout de passerelles (expressions s'interprétant comme $I[c_1]\cap\ldots\cap I[c_n]\subseteq I[c]$). Ce genre d'extension ne semble pas poser de problème particulier. Sa révision a été traitée en temps polynomial dans un cadre cependant plus restreint (pas de types ni d'attributs) [Theodoratos95b].

Enfin, l'intégration à TROPES de problèmes de satisfaction de contraintes nécessitera des compléments autant pour en tenir compte lors du processus de révision que pour proposer des modifications portant sur les contraintes elles-mêmes; cela posera d'importants problèmes de complexité habituels dans ce contexte.

Annexes

A. NOTATIONS 187
1. Opérations sur les types *187*
2. Ensembles d'entités *187*
3. Assertions *187*
4. Interprétation et déduction *188*
B. DÉMONSTRATIONS DES PROPOSITIONS 191
1. Interprétation et déduction *192*
2. Définition de l'interprétation élémentaire *193*
3. Correction *205*
4. Inconsistance et incohérence *206*
5. Complétude faible *211*
6. Forme première *215*
7. Base révisée *220*
C. FORME PREMIÈRE, DÉDUCTION, ACCEPTABILITÉ ET COMPLEXITÉ POUR LE LANGAGE SIMPLE 229
1. Détermination de la forme première *229*
2. Déduction *232*
3. Test d'acceptabilité (consistance, cohérence) *233*
4. Révision *236*
D. FORME PREMIÈRE, DÉDUCTION, ACCEPTABILITÉ ET COMPLEXITÉ POUR LE LANGAGE AVEC RÉFÉRENCE À DES INSTANCES 239
1. Détermination de la forme première *239*
2. Déduction *242*
3. Test d'acceptabilité (consistance, cohérence) *243*
4. Révision *245*
5. Comparaison avec d'autres travaux *249*
E. MINIMALITÉ DE L'ALGORITHME 250
1. Les solutions élémentaires *251*
2. Les solutions globales *258*

H. Notations

1. Opérations sur les types

$d \wedge_\tau d'$ intersection des domaines de valeurs d et d'.

$d =_\tau d'$ ($d \neq_\tau d'$) le domaine d est égal au (resp. différent du) domaine d.

$d \leq:_\tau d'$ ($d <:_\tau d'$) le domaine d est inclus (resp. strictement) dans le domaine d'.

$v :_\tau d$ ($\neg v :_\tau d$) la valeur v appartient (resp. n'appartient pas) au domaine d.

$v \leq_\tau v'$ la valeur v est inférieure à la valeur v' (τ étant ordonné).

$v =_\tau v'$ ($v \neq_\tau v'$) la valeur v est égale à (resp. différente de) la valeur v'.

$\perp_\tau$ symbole représentant le domaine vide du type τ.

2. Ensembles d'entités

C_B l'ensemble des classes c introduites dans la base B par une assertion de la forme $c \leq c'$ ou $c.a \subseteq d$.

O_B l'ensemble des instances o introduites dans la base B par une assertion de la forme $o.a = v$ ou $o \in c$.

A_B l'ensemble des attributs a introduits dans la base B par une assertion de la forme $o.a = v$ ou $c.a \subseteq d$.

V_B l'ensemble des valeurs des types introduits dans la base B.

T_B l'ensemble des domaines des types introduits dans la base B.

3. Assertions

$c \leq c'$ la classe c est sous-classe de la classe c'.

$o \in c$ l'instance o appartient à la classe c.

$c.a \subseteq d$ l'attribut a de la classe c est restreint au domaine d.

$o.a = v$ la valeur de l'attribut a de l'instance o est la valeur v.

$c.a \subseteq c'$ l'attribut a de la classe c est restreint à la classe c'.

$o.a = o'$ la valeur de l'attribut a de l'instance o est l'instance o'.

4. Interprétation et déduction

$<D,I>$ une interprétation est un couple composé d'un domaine et d'une fonction d'interprétation.

$\models_M B$ M est un modèle de la base B.

$B \models \delta$ δ est une conséquence (sémantique) de B.

$B \vdash \delta$ la base B permet de déduire (syntaxiquement) l'assertion δ.

$\mathrm{Cn}(B)$ clôture déductive de la base par les règles d'inférence.

$\mathrm{FP}(B)$ forme première de la base B.

$B' \propto B''$ la base B' est plus petite que la base B''

Sommaire des définitions

Voici le sommaire des définitions données dans le corps principal du mémoire.

DÉFINITION 1 : GRAMMAIRE 51

DÉFINITION 2 : ENSEMBLES D'ENTITÉS 53

DÉFINITION 3 : INTERPRÉTATION DE TYPE 54

DÉFINITION 4 : INTERPRÉTATION DES ENTITÉS D'UNE BASE 55

DÉFINITION 5 : SATISFACTION D'UNE ASSERTION 57

DÉFINITION 6 : MODÈLE 58

DÉFINITION 7 : CONSÉQUENCE 60

DÉFINITION 8 : BASE INCONSISTANTE 60

DÉFINITION 9 : RÈGLES D'INFÉRENCE 61

DÉFINITION 10 : CLÔTURE D'UNE BASE 62

DÉFINITION 11 : DÉDUCTION 62

DÉFINITION 12 : FORME PREMIÈRE 66

DÉFINITION 13 : FORME NORMALE 66

DÉFINITION 14 : GRAMMAIRE 72

DÉFINITION 15 : RÉFÉRENCE SANS CIRCUIT 73

DÉFINITION 16 : INTERPRÉTATION DES ENTITÉS D'UNE BASE 74

DÉFINITION 17 : SATISFACTION D'UNE ASSERTION 75

DÉFINITION 18 : RÈGLES D'INFÉRENCE 77

DÉFINITION 19 : FORME NORMALE (LANGAGE COMPLEXE) 79

DÉFINITION 20 : CLASSE INCOHÉRENTE 86

DÉFINITION 21 : BASE INCOHÉRENTE 86

DÉFINITION 22 : BASE ACCEPTABLE, BASE INACCEPTABLE 87

DÉFINITION 23 : ASSERTION ACCEPTABLE/INACCEPTABLE DANS UNE BASE 87

DÉFINITION 24 : MOTIF D'INCONSISTANCE (LANGAGE SIMPLE) 87

DÉFINITION 25 : MOTIF D'INCONSISTANCE (LANGAGE COMPLEXE).....90

DÉFINITION 26 : ASSERTION COHÉRENTE 95

DÉFINITION 27 : BASE CONTRACTÉE.....96

DÉFINITION 28 : BASE RÉVISÉE 96

DÉFINITION 29 : ORDRE ENTRE LES BASES CONTRACTÉES.....100

DÉFINITION 30 : BASE CONTRACTÉE MINIMALE 100

Les propositions identiques dans le langage complexe et dans le langage simple n'ont pas été redonnées dans le texte; leur numéro correspond donc à celui donné dans le langage simple. De plus, l'ordre a été modifié à cause des dépendances existant entre les démonstrations (en particulier l'inconsistance a été traitée avant la complétude faible). Les démonstrations relatives à la complexité sont données en annexe C et D.

L'index des démonstrations des propositions est donné ci-dessous :

PROPOSITION 1 ET PROPOSITION 15 192
PROPOSITION 2 192
PROPOSITION 3 192
PROPOSITION 4 193
PROPOSITION 5 : COMPLEXITÉ DE LA DÉDUCTION 233
PROPOSITION 6 : CORRECTION 205
PROPOSITION 7 : COMPLÉTUDE FAIBLE 214
PROPOSITION 8 (ET 17) : FORME PREMIÈRE 215
PROPOSITION 9 : UNICITÉ DE LA FORME PREMIÈRE 219
PROPOSITION 10 219
PROPOSITION 11 219
PROPOSITION 12 220
PROPOSITION 13 220
PROPOSITION 14 : COMPLEXITÉ DU CALCUL DE LA FORME PREMIÈRE 232
PROPOSITION 16 193
PROPOSITION 18 : INCONSISTANCE SYNTAXIQUE 206
PROPOSITION 19 ET 20 210
PROPOSITION 21 : CARACTÉRISATION SYNTAXIQUE DE LA MINIMALITÉ 221
PROPOSITION 22 : CARACTÉRISATION SÉMANTIQUE DE LA MINIMALITÉ 220
PROPOSITION 23 : INTERPRÉTATION "OBJET" DE LA MINIMALITÉ 221
PROPOSITION 24 : SATISFACTION DES POSTULATS AGM 228

I. Démonstrations des propositions

Ci-dessous sont données les preuves des propositions pour le langage avec référence aux instances et pour la révision. Les preuves, plus simples, concernant le langage sans référence aux instances sont dans [Crampé&96b].

1. Interprétation et déduction

PROPOSITION 1 ET PROPOSITION 15 :

Si $M=<D,I_B>$ est un modèle de B, alors :

$$I_B(c) \subseteq D \cap \bigcap_{(c \leq c') \in B} I_B(c') \cap \bigcap_{(c.a \subseteq d) \in B} \{i \in D; I_B(a)(i) \in I_B(d)\}$$

$$I_B(o) \in \bigcap_{(o \in c) \in B} I_B(c) \cap \bigcap_{(o.a=v) \in B} \{i \in D; I_B(a)(i) = I_B(v)\}$$

Les fonctions dans $I(T_A^B)$ sont covariantes, c'est-à-dire que si $I(c) \subseteq I(c')$ alors $I(a|c) \subseteq I(a|c')$.

$$I_B(a \mid c) \subseteq (\tau \cup D) \cap \bigcap_{(c \leq c') \in B} I_B(a \mid c') \cap \bigcap_{(c.a \subseteq d) \in B} I_B(d)$$

$$I_B(a)(I_B(o)) \in \bigcap_{(o \in c) \in B} I_B(a \mid c) \cap \bigcap_{(o.a=v) \in B} \{I_B(v)\}$$

démonstration. immédiate vu les contraintes qui pèsent sur les fonctions d'interprétation. ◊

Ci-dessous sont démontrées les propositions introduites avec le système déductif.

PROPOSITION 2

La relation entre bases $\vdash$ est transitive.

démonstration. Soient B, B' et B'' telles que $B \vdash B'$ et $B' \vdash B''$. $\forall \delta \in B''$; $B' \vdash \delta$, alors soit $\delta \in B'$ (et par conséquent $B \vdash \delta$); soit $\delta \notin B'$ et est donc déduite de B' par application des règles ci-dessus à partir d'un ensemble d'assertions appartenant à B'. Or, cet ensemble d'assertions de B' peut être déduit par application des règles à partir d'un ensemble d'assertions appartenant à B. Par conséquent, l'application des règles permet de déduire δ à partir de B et donc $B \vdash B''$. ◊

PROPOSITION 3

Soient B et B' deux bases, $B \vdash B'$ ssi $B \vdash \mathrm{Cn}(B')$.

démonstration.

si) $B' \subseteq \mathrm{Cn}(B')$.

seulement si) $B' \vdash \mathrm{Cn}(B')$ et $\vdash$ est transitive. ◊

PROPOSITION 4

$$(c \leq^* c') \in B \text{ ssi } B \vdash c \leq c'.$$

démonstration.

si : vu la définition de $(c \leq^* c')$, la règle de ($\leq$-réflexivité) s'applique tout le long de la chaîne entre c et c' et on obtient $B \vdash c \leq c'$.

seulement si : l'assertion $c \leq c'$ ne peut s'obtenir que par la règle de ($\leq$-réflexivité) donc par récurrence on montre que soit $(c \leq c') \in B$, soit $B \vdash c \leq c''$ et $B \vdash c'' \leq c'$ avec $(c \leq^* c'') \in B$ et $(c'' \leq^* c') \in B$, d'où $(c \leq^* c') \in B$ par définition de la notation. ◊

PROPOSITION 16

S'il existe $o \in O$, a_1, $a_2, \ldots, a_n \in A$, $o' \in O$, $c' \in C$ tels que $(c.a_1.a_2.\ldots.a_n \subseteq^* c') \in B$, $(o.a_1.a_2.\ldots.a_n =^* o') \in B$, $(o \in c) \in B$ alors $B \vdash o' \in c'$.

démonstration.

Montrons le résultat par récurrence :

$n=1$: $(o.a_1 = o') \in B$, $(c \leq c'') \in B$ et $(c''.a_1 \subseteq c') \in B$. En appliquant la règle de (d-héritage), $B \vdash c.a_1 \subseteq c'$ et en appliquant la règle de ($\in$-inférence) $B \vdash o' \in c'$.

$n>1$: il existe $o' \in O$ tel que $(o.a_1.a_2.\ldots.a_{n-1} =^* o'') \in B$ et $(o''.a_n = o') \in B$ et il existe $c'' \in C$ tel que $(c.a_1.a_2.\ldots.a_{n-1} \subseteq^* c'') \in B$ $(c''.a_n \subseteq^* c') \in B$. Par hypothèse de récurrence, $B \vdash o'' \in c''$. En appliquant la règle de (d-héritage), $B \vdash c''.a_n \subseteq c'$ et en appliquant la règle de ($\in$-inférence) $B \vdash o' \in c'$. ◊

2. Définition de l'interprétation élémentaire

L'interprétation élémentaire correspond à l'interprétation naturelle pour une base, en étant la plus réduite tout en étant un modèle si et seulement si la base est consistante. A chaque instance de la base est associé un objet du domaine, à chaque classe est associée l'ensemble des instances appartenant à cette classe dans la base (ainsi que quelques instances supplémentaires permettant de détecter les classes incohérentes), à chaque attribut est associée une fonction dans le domaine. L'interprétation élémentaire se définit en étendant un noyau d'interprétation. Elle permet de donner une nouvelle caractérisation de base inconsistante ou incohérente. Les interprétations élémentaires qui sont des modèles de la base lorsque celle-ci

est consistante seront utilisées dans la preuve de complétude du système déductif en tant que contr'exemple dans les preuves par l'absurde.

L'interprétation élémentaire doit satisfaire des critères supplémentaires afin d'être exploitable avec profit :

- L'interprétation d'une classe est vide si et seulement si elle est incohérente;
- Tous les ensembles de classes qui sont compatibles (c'est-à-dire qui ne posent pas de restriction contradictoire sur leurs attributs, une instance pouvant donc appartenir à toutes ces classes simultanément) doivent posséder une instance en commun;

et ces contraintes doivent être satisfaites sans enfreindre le fait que l'interprétation d'une classe est contenue dans l'interprétation de ses super-classes.

a) Définition d'ensembles utiles

Certains ensembles vont faciliter la définition de l'interprétation élémentaire :

- L'ensemble des instances attachées à une classe regroupent toutes les instances attachées à la classe directement ou à l'une de ses sous-classes et toutes les instances dont l'attachement est déductible par la règle ($\in$-inférence).
- Un ensemble de classes est incompatible si les restrictions posées sur leurs attributs ne sont pas compatibles.
- L'ensemble des classes atteintes par une instance pour un certain chemin d'attributs regroupe toutes les classes qui sont accessibles par ce chemin d'attribut à partir d'une classe d'appartenance de l'instance. Ces ensembles permettront de détecter les ensembles de classes dans lesquelles il faut mettre une instance commun; de ces ensembles, seuls seront gardés ceux qui regroupent le plus grand ensemble de classes, en considérant l'ensemble des instances et des chemins d'attributs.

DÉFINITION 1' : ensemble des instances attachées à une classe
Soit $c \in C$, l'ensemble des instances attachées à la classe c est :

$$\mathrm{OA}(c)=\{o\ ;\ (o \in c) \in B\} \cup \bigcup_{(c' \leq^* c) \in B} OA(c')$$

$\cup \{o \in O$; il existe $a_1, a_2, \ldots, a_n \in A$, $n \geq 1$, $o' \in O$; $c' \in C$ tels que $(o' \in c') \in B$, $(o'.a_1.a_2.\ldots.a_{n=*}o) \in B$ et $(c'.a_1.a_2.\ldots.a_n \subseteq^* c) \in B$ $\}$

Remarque : cette définition n'est pas récursive car il n'existe pas de cycle dans les chaînes de références aux classes.

DÉFINITION 2' : ENSEMBLE DE CLASSES INCOMPATIBLE

L'ensemble des classes $c_1, c_2, \ldots, c_k \in C$ est incompatible s'il existe $a_1, a_2, \ldots, a_n \in A$, $n \geq 1$, tels que pour tout $1 \leq j \leq k$, $(c_j.a_1.a_2.\ldots a_n \subseteq^* d_l) \in B$, $d_l \in T$ et $\bigwedge_{1 \leq j \leq k} \left(\bigwedge_{d_l ; c_j.a_1.\ldots.a_n \subseteq^* d_l} d_l \right) = \bot$.

DÉFINITION 3' : ENSEMBLE DES CLASSES ATTEINTES PAR UNE INSTANCE

Soient $i \in O$ $a_1, a_2, \ldots, a_n \in A$, $n \geq 1$, l'ensemble des classes atteintes par l'instance i pour la chaîne d'attributs $a_1, a_2, \ldots, a_n$ est :

$n=1$, $CA(i,a_1)=\{c \in C$; il existe $c', c'' \in C$ tel que $(c'' \leq^* c) \in B$, $(c'.a_1 \subseteq^* c'') \in B$, $(i \in c') \in B\}$

$n \geq 2$, $CA(i,a_1, a_2, \ldots, a_n)=\{c \in C$; il existe $c' \in CA(i,a_1, a_2, \ldots, a_{n-1})$, $c'' \in C$ tels que $(c'' \leq^* c) \in B$, $(c'.a_n \subseteq c'') \in B\}$

Par extension de la notation, $CA(i)=\{c \in C$; il existe $c' \in C$ tel que $(c' \leq^* c) \in B$, $(i \in c) \in B\}$

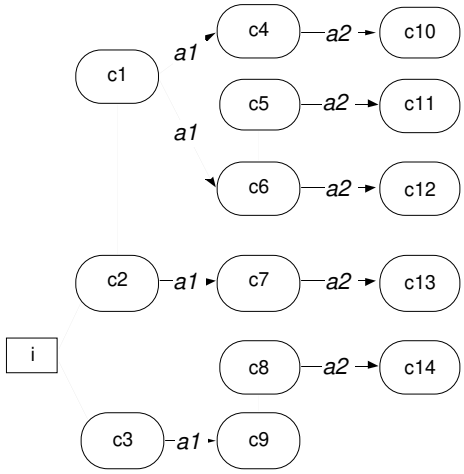


Figure 73 : $CA(i,a_1)=\{c_4, c_6, c_7, c_9, c_5, c_8\}$; $CA(i,a_1, a_2)=\{c_{10}, c_{11}, c_{12}, c_{13}, c_{14}\}$; $CA(i)=\{c_1, c_2, c_3\}$.

Pour l'ensemble des instances et des attributs de la base, ces ensembles $CA(i,a_1, a_2, \ldots, a_n)$ ne sont pas forcément disjoints et, en particulier, certains de ces ensembles en incluent d'autres. Or, ces ensembles sont utiles pour que, dans l'interprétation élémentaire, la même instance arbitraire appartienne à chaque classe d'un même ensemble. Il suffit donc de considérer les maxima de tels ensembles, noté maxCA; en cas d'égalité entre deux ensembles, un seul des ensembles, choisi arbitrairement, appartiendra à maxCA. Cet ensemble maxCA dépend bien entendu de la base, mais il est toujours possible de déduire la base dont il s'agit du contexte.

maxCA={CA($i,a_1, a_2, \ldots, a_n$) tel que $n\geq 0$, $i\in O$, $a_1, a_2, \ldots, a_n\in A$ et quels que soient $k\geq 0$, $i'\in O$ et $a'_1, a'_2, \ldots, a'_k\in A$, CA($i,a_1, a_2, \ldots, a_n$)$\not\subset$CA($i',a'_1, a'_2, \ldots, a'_k$)}

La même instance arbitraire (noté $o_{i,a1, a2, \ldots, an}$ dans la définition suivante) est ajouté dans chacune des classes de CA($i,a_1, a_2, \ldots, a_n$), si cet ensemble est maximum et si aucune instance de la base n'appartient à toutes ces classes; ces conditions assurent que le nombre minimum d'instance est ajouté dans les classes.

b) Conditions nécessaires d'inconsistance

Les quatre lemmes ci-dessous donnent des ensembles d'assertions qui rendent une base inconsistante.

LEMME 1

S'il existe $o\in O$, $a\in A$, $v\in V\cup O$, $v'\in V\cup O$ tels que $(o.a=v)\in B$ et $(o.a=v')\in B$ avec $v\neq v'$, alors la base est inconsistante.

démonstration. Toute fonction d'interprétation I est telle que $I(a)$ est une fonction, ce qui n'est pas possible si $I(a)(I(o))=I(v)$ et $I(a)(I(o))=I(v')$ car I est injective (si $v\neq v'$, alors $I(v)\neq I(v')$). Il ne peut donc pas exister de modèle de B. ◊

LEMME 2

Soient $o\in O$, $a_1, a_2,\ldots,a_n\in A$, $v\in V$, $d\in T$, si $(c.a_1.a_2.\ldots.a_n\subseteq *d)\in B$, $(o.a_1.a_2.\ldots.a_n=*v)\in B$ $(o\in c)\in B$ et $\neg v{:}d$ alors la base est inconsistante.

démonstration. tout modèle doit satisfaire : $I(c_j)\subseteq I(c'_j)$, $I(a_j)I(c_j)\subseteq I(c_{j+1})$ et $I(a_j)(I(o_j))=I(o_{j+1})$; par récurrence, on montre que $I(o_j)\in I(c'_j)$; d'où $I(o_n)\in I(c_n)$, et comme $I(a_n)(I(o_n))=I(v)$, $I(a_n)I(c'_n)\subseteq I(d)$ et $I(v)\notin I(d)$, la base est inconsistante.

variante de la démonstration De la définition de $(c.a_1.a_2.\ldots.a_n\subseteq *d)$ et de $(o.a_1.a_2.\ldots.a_n=*v)$, on peut déduire que tout modèle doit satisfaire : $I(a_n)(I(a_{n-1})(\ldots I(a_1)(I(o))\ldots)=I(v)$ et $I(a_n)(I(a_{n-1})(\ldots I(a_1)(I(c))\ldots)\subseteq I(d)$. De plus, $(o\in c)\in B$ donc $I(o)\in I(c)$, d'où $I(v)\in I(d)$, ce qui est contradictoire avec $\neg v{:}d$. Donc il ne peut exister de tel modèle ◊

Si les classes atteintes par un chemin d'attributs à partir d'une instance sont incompatibles, la base est inconsistante (car il n'existe pas de valeur possible pour l'attribut de l'instance référencée par l'instance référencée par, etc.).

LEMME 3 :

S'il existe $i \in O$ tel qu'un sous-ensemble de CA(i,a_1, a_2, …, a_n) soit un ensemble de classes incompatibles alors la base est inconsistante.

démonstration. soient c_1, c_2, …, $c_k \in$ CA(i,a_1, a_2, …, a_n); les classes sont incompatibles, donc il existe a'_1, a'_2, …, $a'_m \in A$ tels que $(c_j.a'_1.a'_2.\ldots a'_m \subseteq^* d_1) \in B$, $d_1 \in T$ et $\bigwedge_{1 \le j \le m} \left(\bigwedge_{d_l ; c_j.a_1.\ldots.a_n \subseteq^* d_l} d_l \right) = \perp$; tout modèle doit satisfaire $I(a_1(\ldots(I(a_n)(I(i)\ldots) \in I(c_j)$ et donc $I(a'_1(\ldots(I(a'_m(I(a_1(\ldots(I(a_n(I(i)\ldots) \in I(d_1)$. Or, $\bigcap_{\substack{d_l ; c_j.a_1.\ldots.a_n \subseteq d_l \\ 1 \le j \le k}} I(d_l) = \emptyset$, donc une telle interprétation ne peut pas exister. ◊

Si une instance est attachée à un ensemble de classes qui ne sont pas compatibles, la base est inconsistante, car au moins un de ses attribut ne pourra pas être valué.

LEMME 4 :

S'il existe $o \in O$ tel qu'un sous-ensemble de $\{c;\ o \in \text{OA}(c)\}$ soit un ensemble de classes incompatibles alors la base est inconsistante.

démonstration. soient c_1, c_2, …, $c_k \in \{c;\ o \in \text{OA}(c)\}$, les classes incompatibles. Il existe a_1, a_2, …, $a_n \in A$ tels que $(c_j.a_1.a_2.\ldots a_n \subseteq^* d_1) \in B$, $d_1 \in T$ et $\bigwedge_{1 \le j \le k} \left(\bigwedge_{d_l ; c_j.a_1.\ldots.a_n \subseteq^* d_l} d_l \right) = \perp$. D'autre part, $o \in \text{OA}(c_j)$, donc tout modèle satisfait $I(o) \in I(c_j)$; en effet, d'après la définition de OA :

si $(o \in c_j) \in B$, $I(o) \in I(c_j)$;

s'il existe a'_1, a'_2, …, $a'_m \in A$, $m \ge 1$, $o' \in O$; $c'_j \in C$ tels que $(o' \in c'_j) \in B$, $(o'.a'_1.a'_2.\ldots a'_m =^* o) \in B$ et $(c'_j.a'_1.a'_2.\ldots a'_n \subseteq^* c_j) \in B$, par récurrence on montre que $I(o) \in I(c_j)$.

si $(o \in c'_j) \in B$, $(c'_j \le^* c_j) \in B\}$, $I(o) \in I(c'_j) \subseteq I(c_j)$;

s'il existe a'_1, a'_2, …, $a'_m \in A$, $m \ge 1$, $o' \in O$; c'_j, $c_j \in C$ tels que $(c'_j \le^* c_j) \in B$ $(o \in c'_j) \in B$, $(o'.a'_1.a'_2.\ldots a'_m =^* o) \in B$ et $(c'_j.a'_1.a'_2.\ldots a'_m \subseteq^* c_j) \in B$, par récurrence on montre que $I(o) \in I(c'_j) \subseteq I(c_j)$.

Vu la définition de $(c_j.a_1.a_2.\ldots a_n \subseteq *d_1) \in B$, on montre par récurrence que $I(a_1(\ldots(I(a_m(I(o)\ldots) \in I(d_1)$. Or, $\bigcap_{\substack{d_l ; c_j.a_1.\ldots.a_n \subseteq d_l \\ 1 \le j \le k}} I(d_l) = \emptyset$, donc une telle interprétation ne peut pas exister. ◊

c) Noyau d'interprétation

Nous allons définir à présent le noyau d'interprétation qui regroupe ce qui est vérifié par tous les modèles :

- à une instance de la base est associé un objet du domaine;
- à une classe est associé l'ensemble des instances attachées à cette classe ou à l'une de ses sous-classes, ainsi que les instances référencées par un attribut d'une instance appartenant à une classe dont l'attribut référence la classe (les instances appartenant à OA(c)). De plus, une instance arbitraire est ajoutée dans toutes les classes qui font partie d'un ensemble de classes atteintes lorsqu'aucune instance n'appartient à toutes les classes de l'ensemble;
- à un attribut d'instance est associée l'interprétation de sa valeur dans la base (s'il possède une telle valeur);
- à un attribut de classe est associé le domaine résultant de l'intersection des différentes contraintes posées (par ses super-classes ou par les restrictions de la classe elle-même).

DÉFINITION 4' : noyau d'interprétation

K_B est un noyau d'interprétation d'une base B sur un domaine D_K si

$K_B(v)=I(v)$ si $v \in V$

$K_B(d)=I(d)$ si $d \in T$

K_B: $O \rightarrow D_K$ bijective

K_B: $C \rightarrow 2^{D_K}$

$c \rightarrow \{K_B(o); o \in \mathrm{OA}(c)\}$

$\cup \{o_{i,a1}, o_{a2}, \ldots, o_{an}$; $i \in O, a_1, a_2, \ldots, a_n \in A, c \in \mathrm{CA}(i,a_1, a_2, \ldots, a_n), \mathrm{CA}(i,a_1, a_2, \ldots, a_n) \in \mathrm{maxCA}, \bigcap_{c' \in \mathrm{CA}(i,a_1,a_2,\ldots,a_n)} OA(c') = \emptyset$ et pour tout $a \in A$, $\bigcap_{c' \in \mathrm{CA}(i,a_1,a_2,\ldots,a_n)} K_B(a \mid c') \neq \emptyset\}$

K_B: $A \rightarrow (D_K \rightarrow \tau \cup D)$ est une fonction partielle

$K_B(a)$: $(K_B(c) \rightarrow (\tau \cup D) \cap \bigcap_{(c.a \subseteq d_i) \in B} K_B[d_i] \cap \bigcap_{(c \le c') \in B} K_B(a|c'))$

$K_B(a)(K_B(o)) = K(v)$ si $(o.a=v) \in B$.

REMARQUES :

- Cette définition est bien fondée, malgré la récursivité introduite par $K_B(a|c)$ (qui utilise à son tour $K_B(c)$) dans $K_B(c)$. En effet, comme il n'y a pas de référence circulaire, il est possible de commencer à définir $K_B(c)$ pour les classes qui ne sont pas atteintes.
- K_B ne constitue pas nécessairement une interprétation (car $K_B(a)$ n'est pas définie sur tout l'ensemble $\{K_B(o), o \in O\}$), mais la notation $K_B(a|c)$ sera utilisée comme pour les fonctions d'interprétation : $K_B(a|c)=(\tau \cup D) \cap \bigcap_{\substack{(c'.a \subseteq d) \in B \\ (c \leq^* c') \in B}} K_B(d)$
- Le noyau d'interprétation dépend de la bijection initiale de O vers D_K. L'ensemble des noyaux d'interprétation constitue une classe de fonctions équivalentes à un isomorphisme sur D_K près. On considérera dorénavant un unique K_B.

d) Interprétation élémentaire

Pour définir la fonction d'interprétation, dite élémentaire et notée $\hat{I}$, il reste alors à :

- ajouter une instance arbitraire (noté o_c dans la définition suivante) dans toutes les classes, cette instance satisfaisant les contraintes posées sur les attributs de la classe (ceci permet d'une part de distinguer les cas de classe incohérente de ceux de classes sans instances et d'autre part de pouvoir assurer que $\hat{I}(c) \subseteq \hat{I}(c')$ seulement si $(c \leq^* c') \in B$)
- donner une valeur à tous les attributs des instances qui n'ont pas de valeur définie dans la base, en respectant les contraintes posées par les classes d'appartenance de l'instance.

DÉFINITION 5' : interprétation élémentaire

$<D,\hat{I}>$ est une interprétation élémentaire de B si $\hat{I}$ est une fonction qui étend K_B un noyau d'interprétation sur D_K, de la manière suivante :

$\hat{I}(o) = K_B(o)$

$\hat{I}(a|c) = K_B(a|c)$

$\hat{I}(c) = \emptyset$ si $\exists a, \hat{I}(a|c)=\emptyset$;

$\{o_c\} \cup K_B(c) \cup \bigcup_{(c' \leq c) \in B} \hat{I}(c')$ sinon

$\hat{I}(a)(\hat{I}(o))= K_B(a)(K_B(o))$ si défini

$\in \bigcap_{\hat{I}(o) \in K_B(c)} \hat{I}(a|c)$ sinon

$\hat{I}(a)(o_c) \in \hat{I}(a|c)$

$\hat{I}(a)(o_{i,a1}, {}_{a2}, \ldots, {}_{an}) \in \bigcap_{o_{i,a_1}, a_2, \ldots, a_n \in K_B(c)} \hat{I}(a \mid c)$

D est l'ensemble D_K augmenté de l'ensemble des instances o_c et $o_{i,a}$ nécessaires pour la définition.

Il existe en général plusieurs interprétations élémentaires, étendant un même noyau d'interprétation, car les valeurs des attributs des instances (et en particulier des instances introduites dans la définition et notés o_c ou $o_{i,a1}, {}_{a2}, \ldots, {}_{an}$) ne sont pas nécessairement déterminées. Par abus de notation, toute interprétation élémentaire sera notée $<D,\hat{I}>$.

e) Modèle élémentaire

Nous allons montrer que l'interprétation élémentaire est modèle de la base si celle-ci est consistante. Pour ce faire, nous donnons tout d'abord une caractérisation de l'inconsistance. La proposition suivante indique qu'il y a inconsistance si les restrictions posées sur un attribut d'un ensemble de classes d'appartenance d'une instance sont incompatibles.

Un cas particulier de cette proposition est celui où la restriction d'attribut pour une classe possédant des instances est vide.

LEMME 5 :

S'il existe $o' \in D_K$, $a \in A$ tels que $\bigcap_{c;\, o' \in K_B(c)} K_B(a \mid c) = \emptyset$, alors il existe une instance o attachée à un ensemble de classes incompatibles et la base est inconsistante.

démonstration. La démonstration se décompose en deux parties selon que o' peut s'écrire $K_B(o)$ ou $o_{i,b}$.

S'il existe o tel que $o' = K_B(o)$, il s'agit de montrer que les classes d'appartenance de o forment un ensemble incompatible, sachant que $\{c;\, K_B(o) \in K_B(c)\} = \{c;\, o \in \mathrm{OA}(c)\}$.

$$\bigcap_{K_B(o) \in K_B(c)} K_B(a|c) = \bigcap_{c;\, K_B(o) \in K_B(c)} \left(\bigcap_{d;\, (c.a \subseteq^* d) \in B} K_B(d) \right)$$

Si tout d, $d \in T$ alors le résultat est prouvé, d'après la Définition 2' d'un ensemble de classes incompatibles (car l'intersection des interprétations des domaines est égale à l'interprétation de l'intersections des domaines et celle-ci n'est vide que si l'intersection des domaines est vide).

Si pour tout d, $d \in C$ alors $d \in \mathrm{CA}(o,a)$ et il existe $i \in O$, $b \in A$ tels que $\mathrm{CA}(o,a) \subseteq \mathrm{CA}(i,b)$ et $\mathrm{CA}(i,b) \in \mathrm{maxCA}$ (car $\mathrm{CA}(o,a)$ n'est pas forcément un ensemble maximal). Donc $o_{i,b} \in K_B(d)$, quel que soit $K_B(d)$ terme de l'intersection, ce qui est contradictoire

avec le fait que l'intersection est vide, donc il existe $a_2 \in A$ tel que $\bigcap_{c;\, K_B(o)\in K_B(c)} \left(\bigcap_{d;\,(c.a\subseteq^* d)\in B} K_B(a_2 \mid d) \right) = \emptyset = \bigcap_{c;\, K_B(o)\in K_B(c)} \left(\bigcap_{d;\,(c.a.a_2\subseteq^* d)\in B} K_B(d) \right) = \emptyset$. De la même manière que précédemment, il existe $a_3 \in A$ tel que $\bigcap_{c;\, K_B(o)\in K_B(c)} \left(\bigcap_{d;\,(c.a.a_2\subseteq^* d)\in B} K_B(a_3 \mid d) \right) = \bigcap_{c;\, K_B(o)\in K_B(c)} \left(\bigcap_{d;\,(c.a.a_2.a_3\subseteq^* d)\in B} K_B(d) \right) = \emptyset$ car la définition d'une instance $o_{\mathrm{i,b,b2,b3}}$ appartenant à chacun des $K_{\mathrm{B}}(d)$ n'est pas possible.

Par récurrence, on montre donc que $\bigcap_{c;\, K_B(o)\in K_B(c)} \left(\bigcap_{d;\,(c.a.a_2 \ldots .a_n\subseteq^* d)\in B} K_B(d) \right) = \emptyset$ avec $d \in T$, c'est-à-dire que les classes sont incompatibles (cette récurrence est finie, car le nombre de classes est finie et qu'il n'existe pas de cycle dans les références).

Si $o' = o_{\mathrm{i,b}}$, $\bigcap_{o_{i,b}\in K_B(c)} K_B(a|c) = \bigcap_{c\in CA'(i,b)} K_B(a|c) = \emptyset$. On a alors $\bigcap_{c\in CA(i,b)} K_B(c) = \emptyset$, car si $\bigcap_{c\in CA(i,b)} K_B(c) \neq \emptyset$, il existe $o \in D_{\mathrm{K}}$ tel que $o \in K_{\mathrm{B}}(c)$ pour $c \in CA(i,b)$ et donc $K_{\mathrm{B}}(a)(K_{\mathrm{B}}(o)) \in \bigcap_{c\in CA'(i,b)} K_B(a|c)$, ce qui est impossible. D'où $\bigcap_{(i\in c'')\in B} \left(\bigcap_{\substack{(c'.a\subseteq c)\in B \\ (c''\leq^* c')\in B}} K_B(c) \right) = \emptyset$ et on se ramène au cas précédent.

D'après le Lemme 4, si une instance est attachée à un ensemble de classes incompatibles, la base est inconsistante. ◊

La proposition suivante indique que si l'interprétation de la valeur d'un attribut d'une instance n'appartient pas à l'interprétation de l'attribut de la classe, alors soit les classes d'appartenance de l'instance forment un ensemble de classes incompatibles, soit une chaîne d'attribut à partir de l'objet mène à une valeur qui n'appartient pas à la restrcition d'attribut d'une classe obtenue par cette même chaîne.

A noter qu'il suffit que l'instance référencée satisfasse les contraintes posées sur la classe pour avoir consistance, il n'est pas nécessaire que cette instance soit explicitement attachée à la classe ou à l'une de ses sous-classes.

LEMME 6 :

S'il existe $o \in O$, $a \in A$, $v \in V \cup \boldsymbol{O}$ tels que $(o.a=v) \in B$, $(o \in c) \in B$ et $I(v) \notin K_B(a|c)$, alors l'une des deux propositions ci-dessous est satisfaite :

(1) l'ensemble des classes d'appartenance de l'instance o est incompatible

(2) il existe $a_1, a_2, \ldots, a_n \in A$, $v' \in V$, $d \in T$ tels que $(c.a.a_1.a_2.\ldots.a_n \subseteq *d) \in B$, $(v.a_1.a_2.\ldots.a_n = *v') \in B$ et $\neg v:d$

et la base est inconsistante.

démonstration. Comme $I(v) \notin K_B(a|c)$, il existe $c' \in C$ telle que $(c \leq *c') \in B$ telles que $(c'.a \subseteq d) \in B$ et $I(v) \notin K_B(d)$.

Si $v \in V$, $I(v) \notin K_B(d)$ signifie que $\neg v:d$ et on est dans le cas (2) (avec n=1)

Si $v \in O$, $v \in \mathrm{OA}(d)$, par définition de $\mathrm{OA}(d)$; or $I(v) \notin K_B(d)$, ce qui signifie qu'il existe $a_1 \in A$, tel que $\hat{I}(a_1)(\hat{I}(v)) \notin \hat{I}(a_1|d)$ (pour «empêcher» de construire K).

S'il n'existe pas $o_1 \in O$, tel que $(v.a_1=o_1) \in B$, $\hat{I}(a_1|d)=\emptyset$, on est dans le cas (1) (car o appartient à c et $c.a.a_1$ est vide)

S'il existe $o_1 \in O$, tel que $(v.a_1=o_1) \in B$, on prouve le résultat par récurrence : il existe n fini (car le nombre d'objets et le nombre d'attributs sont finis et que les références circulaires ne sont pas autorisées.) tel que $a_1, \ldots, a_n \in A$, $(v.a_1.\ldots.a_n = *v') \in B$ et il n'existe pas $b \in A$, $v \in V \cup O$ tel que $(v.a_1.\ldots.a_n.b = *v') \in B$. Si $v' \in V$, on est dans le cas (2), sinon dans le cas (1).

On peut alors en déduire que la base est inconsistante : par le Lemme 4 pour le cas (1) et par le Lemme 2 pour le cas (2). ◊

LEMME 7 :

Si une base est consistante, les interprétations élémentaires de cette base sont des modèles de la base.

démonstration. Pour qu'un couple $<D,\hat{I}>$ soit un modèle, il faut et il suffit que $<D,\hat{I}>$ soit une interprétation et que la fonction d'interprétation satisfasse toutes les assertions de la base. $<D,\hat{I}>$ est une interprétation aux conditions suivantes :

$\hat{I}$: $O \rightarrow D$ est une fonction injective. Or, $K_B(o) \in D_K \subseteq D$ donc $\hat{I}$ est injective vu qu'elle n'est définie que sur D_K et qu'elle est bijective sur ce sous-ensemble de D.

$\hat{I}$: $C \rightarrow 2^D$. Or, pour tout o, $K_B(o)$ appartient à D, ainsi que tous les éléments notés o_c et $o_{i,a}$.

$\hat{I}$: $A \rightarrow (D \rightarrow \tau \cup D)$: $\hat{I}(a)$ est une fonction totale aux conditions suivantes :

(1) $\hat{I}(a)(o)$ est défini pour chaque $o \in D$.

(2) si $\hat{I}(c)\subseteq\hat{I}(c')$ alors $\hat{I}(a|c)\subseteq\hat{I}(a|c')$

(3) si $o\in\hat{I}(c)$, alors $\hat{I}(a)(o)\in\hat{I}(a|c)$

(1)Pour chaque instance $o\in O$:

Si $K_B(a)(K_B(o))$ est défini, comme la base est consistante, d'après le Lemme 1, il n'existe pas $(o.a=v)\in B$ et $(o.a=v')\in B$ avec $v\neq v'$.

Si $K_B(a)(K_B(o))$ n'est pas défini, comme la base est consistante, d'après le Lemme 5, $\bigcap_{\hat{I}(o)\in K_B(c)}\hat{I}(a|c)\neq\emptyset$, donc il existe $\hat{I}(a)(\hat{I}(o))$.

$\hat{I}(a)(o_c)\in\hat{I}(a|c)$; or par définition de o_c, $\hat{I}(a|c)\neq\emptyset$.

$\hat{I}(a)(o_{i,a1}, a2, \ldots, an)\in \bigcap_{o_{i,a_1},a_2,\ldots,a_n\in K_B(c)}\hat{I}(a\,|\,c)$ est défini si $\bigcap_{o_{i,a_1},a_2,\ldots,a_n\in K_B(c)}\hat{I}(a\,|\,c)\neq\emptyset$; or, cette intersection est non vide d'après le Lemme 5 vu que la base est consistante.

(2) si $\hat{I}(c)\subseteq\hat{I}(c')$ alors si $\hat{I}(c)=\emptyset$, $\hat{I}(a|c)=\emptyset$ et le résultat est trivial; sinon $(c\leq^* c')\in B$. En effet, on a en particulier $o_c\in\hat{I}(c')$, ce qui n'est possible que si $(c\leq^* c')\in B$. D'après la définition de $\hat{I}(a|c)=K_B(a|c)$, on a alors $\hat{I}(a|c)\subseteq\hat{I}(a|c')$.

(3)pour $o_c\in\hat{I}(c)$, $\hat{I}(a)(o_c)\in\hat{I}(a|c)$ par définition.

pour $o_{c'}\in\hat{I}(c)$, alors $c'\in C$, $(c'\leq^* c)\in B$; $\hat{I}(a)(o_c)\in\hat{I}(a|c')\subseteq\hat{I}(a|c)$ (d'après le point (2)).

pour $o_{i,a1}, a2, \ldots, an\in\hat{I}(c)$, $\hat{I}(a)(o_{i,a1}, a2, \ldots, an)\in\hat{I}(a|c)$ (car $\bigcap_{o_{i,a_1},a_2,\ldots,a_n\in K_B(c)}\hat{I}(a\,|\,c)\neq\emptyset$).

pour $\hat{I}(o)\in\hat{I}(c)$, alors :

si $(o.a=v)\notin B$, $\hat{I}(a)(\hat{I}(o))\in\bigcap_{\hat{I}(o)\in K_B(c)}\hat{I}(a|c)$, et cette intersection est non nulle d'après le Lemme 5, donc $\hat{I}(a)(\hat{I}(o))\in\hat{I}(a|c)$

si $(o.a=v)\in B$, comme $o\in\mathrm{OA}(c)$:

si $(o\in c)\in B$, d'après le Lemme 6, $\hat{I}(a)(\hat{I}(o))=\hat{I}(v)\in\hat{I}(a|c)$.

si o est tel qu'il existe $a_1, a_2, \ldots, a_n\in A$, $n\geq 1$, $o'\in O$, $c'\in C$ tels que $(o'\in c')\in B$, $(o'.a_1.a_2.\ldots.a_n=^* o)\in B$ et $(c'.a_1.a_2.\ldots.a_n\subseteq^* c)\in B$; d'où $(o'.a_1.a_2.\ldots.a_n.a=v)\in B$ et pour tout $(c.a\subseteq d)\in B$, $(c'.a_1.a_2.\ldots.a_n.a\subseteq^* d)\in B$. Si $v\in O$, $v\in\mathrm{OA}(d)$ et donc $\hat{I}(v)\in\hat{I}(d)$. Si $v\in V$, comme la base est consistante, pour tout $(c.a\subseteq d)\in B$, $v{:}d$ d'après le Lemme 6 car $(o'\in c')\in B$,

$(c'.a_1.a_2.\ldots.a_n.a \subseteq *d) \in B$ et $(o'.a_1.a_2.\ldots.a_n.a=v) \in B$. Donc $\hat{I}(v) \in \hat{I}(a|c)$ (car $\hat{I}(v)$ appartient à chacun des membres de l'intersection).
s'il existe c' tel que $(o \in c') \in B$ et $(c' \leq c) \in B$, $\hat{I}(a|c') \subseteq I(a|c)$ d'après le point (2) et $\hat{I}(v) \in \hat{I}(a|c')$ d'après le Lemme 6 donc $\hat{I}(a)(\hat{I}(o))=\hat{I}(v) \in \hat{I}(a|c)$.
si o est tel que il existe $c» \in C$ tel que $(c'' \leq *c) \in B$ et $o \in \mathrm{OA}(c»)$; il existe a_1, a_2, …, $a_n \in A$, $n \geq 1$, $o' \in O$, $c' \in C$ tels que $(o' \in c') \in B$, $(o'.a_1.a_2.\ldots.a_n = *o) \in B$ et $(c'.a_1.a_2.\ldots.a_n \subseteq *c'') \in B$. On démontre comme ci-dessus que $\hat{I}(v) \in \hat{I}(a|c'')$ et comme $\hat{I}(a|c'') \subseteq I(a|c)$, $\hat{I}(a)(\hat{I}(o))=\hat{I}(v) \in \hat{I}(a|c)$.

$<D,\hat{I}>$ est donc une interprétation. Par ailleurs, toute interprétation élémentaire satisfait chacun des types possibles d'assertions si la base est consistante. En effet :

$(o \in c) \in B$: la base est consistante donc $\hat{I}(a|c) \neq \emptyset$ (d'après le Lemme 5) et donc $\hat{I}(c) \neq \emptyset$; $\hat{I}(c)$ est donc défini par la deuxième alternative de sa définition et $K_B(c) \subseteq \hat{I}(c)$ avec $K_B(o) \in K_B(c)$, donc $K_B(o)=\hat{I}(o) \in \hat{I}(c)$. De plus, d'après le point (3) ci-dessus, pour tout a $\hat{I}(a)(\hat{I}(o)) \in \hat{I}(a|c)$.

$(c \leq c') \in B$: Si $\hat{I}(c')=\emptyset$, $K_B(a|c')=\emptyset$ et comme $K_B(a|c) \subseteq K_B(a|c')$, $K_B(a|c)=\emptyset$ et $\hat{I}(c)=\emptyset \subseteq \hat{I}(c')$; si $\hat{I}(c') \neq \emptyset$, $\hat{I}(c) \subseteq \hat{I}(c')$ d'après la définition.

$(c.a \subseteq d) \in B$: $K_B(a|c)=\hat{I}(a|c) \subseteq \hat{I}(d)$ par définition de $K_B(a|c)$.

$(c.a \subseteq c') \in B$: idem ci-dessus.

$(o.a=v) \in B$: $\hat{I}(a)(\hat{I}(o))=\hat{I}(v)$, par définition de $\hat{I}(a)$, ce qui est possible car il n'existe pas $(o.a=v) \in B$ et $(o.a=v') \in B$ avec $v \neq v'$, sinon la base est inconsistante (d'après le Lemme 1).

$(o.a=o') \in B$: idem ci-dessus. ◊

La réciproque de cette proposition est immédiate car dès qu'il existe un modèle d'une base, celle-ci est consistante.

VOCABULAIRE

Si la base est consistante, toute interprétation élémentaire est appelé modèle élémentaire.

3. Correction

Nous démontrons à présent la correction du système déductif, présentée au chapitre II.

PROPOSITION 6 : correction
Soient B une base et δ une assertion, si $B \vdash \delta$ alors $B \models \delta$.

démonstration. Soit δ tel que $B \vdash \delta$. $B \models \delta$ peut être prouvé par récurrence sur l'arbre de déduction. Cet arbre a une taille finie car toute déduction se fait en un nombre fini d'application de règles. En effet, les classes et instances d'une base sont en nombre fini et dans les règles, l'utilisation des assertions sur les domaines les plus restreints pour chaque couple classe-attribut est suffisante, donc un nombre fini d'assertions est nécessaire pour calculer chaque élément de Cn(B). La récurrence se fait sur la hauteur (p) de l'arbre développé pour déduire δ.

p=0 : $\delta \in B$ donc $B \models \delta$.

hypothèses de récurrence : pour tout f telle que $B \vdash f$ en utilisant un arbre de hauteur p, $B \models f$.

Soit δ tel que $B \vdash \delta$ en utilisant un arbre de déduction de hauteur $p+1$; la dernière règle de déduction utilisée est de la forme : $\frac{\delta_1 \ldots \delta_n}{\delta}$ avec δ_1, $\ldots \delta_n$ déduits à partir d'un arbre de hauteur au plus p (et dont l'éventuelle condition est vraie). Par hypothèse de récurrence $B \models \delta_1$, … $B \models \delta_n$. Il faut montrer que $B \models \delta$, c'est-à-dire que $\models_M \delta$ pour tout modèle M=<D,I>. La proposition se démontre pour chacune des règles d'inférence :

≤-réflexivité : si $B \vdash c \leq c$, comme $I(c) \subseteq I(c)$, $B \models c \leq c$.

≤-transitivité : si $B \vdash c \leq c''$ avec $B \models c \leq c'$, $B \models c' \leq c''$, alors $\forall M$=<D,I> tel que $\models_M B$, $I(c) \subseteq I(c')$ et $I(c') \subseteq I(c'')$, d'où $I(c) \subseteq I(c'')$ et donc $\forall$ M modèle de B, $\models_M c \leq c''$ d'où : $B \models c \leq c''$.

∈-clôture : si $B \vdash o \in c'$ avec $B \models o \in c$ et $B \models c \leq c'$, alors $\forall M$=<D,I> tel que $\models_M B$, $I(c) \subseteq I(c')$ et $I(o) \in I(c)$, d'où $I(o) \in I(c')$; donc, $\forall M$ modèle de B, $\models_M o \in c'$, et par conséquent $B \models o \in c'$.

d-intersection : si $B \vdash c.a \subseteq d \wedge d'$ avec $B \models c.a \subseteq d$ et $B \models c.a \subseteq d'$ alors $\forall M$=<D,I> tel que $\models_M B$, $I(a|c) \subseteq I(d)$ et $I(a|c) \subseteq I(d')$, d'où $I(a|c) \subseteq I(d) \cap I(d') = I(d \wedge d')$. Donc $\forall M$=<D,I>, $\models_M c.a \subseteq d \wedge d'$ et donc $B \models c.a \subseteq d \wedge d'$.

d-héritage : si $B \vdash c'.a \subseteq d$ avec $B \models c.a \subseteq d$ et $B \models c' \leq c$, alors $\forall M$=<D,I> tel que $\models_M B$, $I(a|c) \subseteq I(d)$ et $I(a|c') \subseteq I(a|c)$ car $I(c') \subseteq I(c)$, d'où $I(a|c') \subseteq I(d)$ $\forall M$ modèle de B, $\models_M c'.a \subseteq d$ donc $B \models c'.a \subseteq d$.

d-clôture : si $B \vdash c.a \subseteq d'$ et $d<:d'$ avec $B \models c.a \subseteq d$, alors $\forall M=<D,I>$ tel que $\models_M B$, $I(a|c) \subseteq I(d)$ et $I(d) \subseteq I(d')$, d'où $I(a|c) \subseteq I(d')$. Par conséquent, $\forall M$ modèle de B, $\models_M c.a \subseteq d'$ et donc $B \models c.a \subseteq d'$.

c-clôture : si $B \vdash c.a \subseteq c''$ avec $B \models c.a \subseteq c'$ et $B \models c' \leq c''$, alors $\forall M=<D,I>$ tel que $\models_M B$, $I(a|c) \subseteq I(c')$ et $I(c') \subseteq I(c'')$, d'où $I(a|c) \subseteq I(c'')$. Par conséquent, $\forall M$ modèle de B, $\models_M c.a \subseteq c''$ et donc $B \models c.a \subseteq c''$.

o-valuation : si $B \vdash o.a=v$ avec pour $i \in [1..n]$ $B \models c_i.a \subseteq d_i$, $B \models o \in c_i$ et $\bigwedge_i d_i = \{v\}$ alors $\forall M=<D,I>$ tel que $\models_M B$, $I(a|c_i) \subseteq I(d_i)$ d'où $I(a)(I(o)) \in I(d_i)$ pour $i \in [1..n]$; donc $I(a)(I(o)) \in I(\bigwedge_i d) = I(v)$. Ce qui signifie que $\forall M$ modèle de B, $\models_M o.a=v$, donc $B \models o.a=v$.

$\in$-inférence : si $B \vdash o' \in c'$ avec $B \models c.a \subseteq c'$, $B \models o \in c$ et $B \models o.a=o'$, alors $\forall M=<D,I>$ tel que $\models_M B$, $I(a|c) \subseteq I(c')$ et comme $I(o) \in I(c)$, $I(a)(I(o)) \in I(a|c)$, d'où $I(a)(I(o))=I(o') \in I(c')$; donc $\forall M$ modèle de B, $\models_M o' \in c'$ donc $B \models o' \in c'$.

Conclusion de la récurrence : $\forall\ \delta$ et p tels que $B \vdash \delta$ en utilisant un arbre de déduction de hauteur p (pour toutes les assertions, il existe un tel arbre de déduction), alors $B \models \delta$. Par conséquent, si $B \vdash \delta$ alors $B \models \delta$. ◊

4. Inconsistance et incohérence

Grâce au Lemme 7 et aux propositions donnant des conditions pour avoir une base inconsistante, il est possible de caractériser syntaxiquement l'inconsistance d'une base. Une inconsistance peut apparaître dans une base si un attribut d'instance :

- possède deux valeurs;
- possède une valeur qui est incompatible avec l'intersection des restrictions pour l'attribut : dans le cas des valeurs typées, il existe un domaine particulier auquel elle n'appartient pas; dans le cas des instances, l'instance référencée n'est pas compatible avec l'intersection des classes(ce qui se ramène au cas suivant);
- ne possède pas de valeur, mais l'ensemble des contraintes portant sur la valeur de l'attribut sont incompatibles. Plus précisément, il existe une instance pour laquelle l'intersection des restrictions d'attribut d'un ensemble de classes dont il est instance directement ou instance de l'une des sous-classes est vide (et donc il n'existe pas de modèle de la base, vu qu'il ne peut y avoir d'interprétation à la valeur d'attribut de l'instance).

Ceci est rassemblé dans la proposition suivante.

LEMME 8 :

Une base B est inconsistante si et seulement si l'une des trois conditions suivantes est vérifiée :

- il existe $o \in O$, $v, v' \in V \cup O$ et $a \in A$ tels que $(o.a=v) \in B$ et $(o.a=v') \in B$ avec $v \neq v'$

- il existe $o\in O$, a_1, $a_2,\ldots,a_n\in A$, $v\in V$, $d\in T$ tels que $(c.a_1.a_2.\ldots a_n\subseteq^* d)\in B$, $(o.a_1.a_2.\ldots a_{n=^*}v)\in B$, $(o\in c)\in B$ et $\neg v{:}d$.[20]
- il existe $o\in O$, c_1, c_2, …, $c_k\in C$, a_1, a_2, …, $a_n\in A$, $n\geq 1$, tels que pour tout $1\leq j\leq k$, $B\vdash(o\in c_j)$, $(c_j.a_1.a_2.\ldots a_n\subseteq^* d_1)\in B$, $d_1\in T$ et $\bigwedge_{1\leq j\leq k}\left(\bigwedge_{d_l;\, c_j.a_1.\ldots.a_n\subseteq^* d_l} d_l\right)=\bot$.

démonstration.

seulement si : La démonstration du Lemme 7 a montré que les seules possibilités d'inconsistance (c'est-à-dire les seuls cas pour lesquels il n'existe pas de modèle) sont celles pointées par les Lemme 1, Lemme 5 et Lemme 6. Le Lemme 1 correspond au premier point. Le Lemme 5 (ensemble de classes incompatibles d'une instance *o*) équivaut au troisième point. Le Lemme 6 quant à elle se ramène au second point (ou au Lemme 5).

si : Le résultat est obtenu par application des Lemme 1, Lemme 2 et Lemme 4 couplé avec les Définition 1' et Définition 3'. ◊

Nous donnons ci-dessous une caractérisation différente qui utilise la déduction. La caractérisation est plus simple dans le cas du langage simple que dans le cas du langage avec référence aux instances, donc elle est donnée uniquement dans ce dernier cas.

PROPOSITION 18: inconsistance syntaxique

B est inconsistante ssi elle contient au moins un motif d'inconsistance, c'est-à-dire l'un des trois ensembles d'assertions suivant :

(1) $\{o.a=v,\ o.a=v'\}$ tel que $(o.a=v)\in B$, $(o.a=v')\in B$ avec $v\neq v$, $o\in O$, $a\in A$, $v,v'\in V\cup\boldsymbol{O}$.

(2) $\{o\in c,\ c.a\subseteq d,\ o.a=v\}$ tel que $B\vdash o\in c$, $B\vdash c.a\subseteq d$, $(o.a=v)\in B$ avec $\neg v:d$, $o\in O$, $c\in C$, $a\in A$, $d\in T$ et $v\in V$.

(3) $\{o\in c_i\}_i\cup\{c_i.a_1.a_2.\ldots a_n\subseteq^* d_{ij}\}_i$ tel que pour tout i, $B\vdash o\in c_i$, $(c_i.a_1.a_2.\ldots a_n\subseteq^* d_{ij})\in B$ avec $\bigwedge_{ij} d_{ij}=\bot$, $o\in O$, $c_i\in C$, $a_i\in A$.

démonstration.

[20]On retrouve le deuxième cas d'inconsistance dans une base sans référence aux objets en choisissant n=1, vu la définition de $(c.a\subseteq^* d)\in B$.

si) Dans chacun des trois cas, il est impossible de trouver un modèle :

1) Si $(o.a=v)\in B$, $(o.a=v')\in B$, alors $\hat{I}(a)(\hat{I}(o))=\hat{I}(v)$, $\hat{I}(a)(\hat{I}(o))=\hat{I}(v')$ et $v\neq v'$ ce qui est contradictoire avec le fait que $\hat{I}$ soit injective. Donc, il ne peut exister de modèle $\hat{I}$.
2) Si $(o.a=v)\in B$, $B\vdash c.a\subseteq d$, $B\vdash o\in c$ et $\neg v:d$, (ou $\neg B\vdash v\in d$) alors, si la base est consistante, $B\models o.a=v$, $B\models c.a\subseteq d$, $B\models o\in c$, $B\not\models v\in d$ et donc $\hat{I}(v)=\hat{I}(a)(\hat{I}(o))\notin\hat{I}(d)$, ce qui n'est pas possible, donc la base est inconsistante.
3) Si $B\vdash o\in c_i$ et $(c_i.a_1.a_2.\ldots a_n\subseteq^* d_i)\in B$, alors $\hat{I}(o)\in\hat{I}(c_i)$ et par récurrence, $\hat{I}(a_n)(\hat{I}(a_{n-1})(\ldots\hat{I}(a_1)(\hat{I}(o))\ldots)\in\bigcap_i\hat{I}[\wedge_i d_i]=\emptyset$, donc $\hat{I}$ n'est pas un modèle.

seulement si) Par le Lemme 8, il existe exactement trois cas d'inconsistance :

1) $(o.a=v)\in B$, $(o.a=v')\in B$ et $v\neq v'$.
2) il existe $o\in O$, a_1, $a_2,\ldots,a_n\in A$, $v\in V$, $d\in T$ tels que $(c.a_1.a_2.\ldots a_n\subseteq^* d)\in B$, $(o.a_1.a_2.\ldots a_n=^* v)\in B$, $(o\in c)\in B$ et $\neg v:d$. Vu la signification des notations, il existe o_1, $o_2,\ldots o_n\in O$ tels que pour tout $1\leq j<n$, $(o_j.a_j=o_{j+1})\in B$ avec $o=o_1$ et $(o_n.a=v)\in B$ et c_1, $c_2,\ldots c_n$, c'_1, $c'_2,\ldots c'_n\in C$ tels que pour tout $1\leq j\leq n$, $(c_i\leq^* c'_i)\in B$, pour tout $1\leq j<n$, $(c'_j.a_j\subseteq c_{j+1})\in B$ avec $c=c_1$ et $(c'_n.a\subseteq d)\in B$. D'après la Proposition 16, $B\vdash o_n\in c_n$; de plus, $B\vdash o_n.a=v$, $B\vdash c_n.a_n\subseteq d$ et $\neg v:d$, ce qui est exprimé dans la deuxième condition.
3) il existe $o\in O$, c_1, c_2, …, $c_k\in C$, a_1, a_2, …, $a_n\in A$, $n\geq 1$, tels que pour tout $1\leq j\leq k$, $(o\in c_j)\in B$, $(c_j.a_1.a_2.\ldots a_n\subseteq^* d_1)\in B$, $d_1\in T$ et $\wedge_{1\leq j\leq k}\left(\wedge_{d_l;\,c_j.a_1.\ldots.a_n\subseteq^* d_l} d_l\right)=\perp$.

$B\vdash o\in c_j$ et $(c_j.a_1.a_2.\ldots a_n\subseteq^* d_1)\in B$, donc le résultat est immédiat. (à noter que l'ensemble $\{c_i\}$ n'est pas forcément unique)

Ces cas se ramènent donc aux expressions données dans la proposition. ◊

Le même type de résultat peut être obtenu pour l'incohérence. Rappelons qu'une classe est incohérente si elle ne peut pas avoir d'instance. Une base possède donc une classe incohérente si la base résultant de l'ajout d'une instance dans cette classe est inconsistante.

Dans le cas du langage simple, une classe est incohérente si son interprétation est vide. Dans le cas du langage avec référence aux instances, une classe est incohérente si un chemin d'attributs mène à un ensemble de domaines dont l'intersection est vide.

PROPOSITION 19 ET 20 :

Soit B une base consistante, $c \in C$ est une classe incohérente si et seulement s'il existe a_1, a_2, …, $a_n \in A$, $n \geq 1$, tels que pour tout $1 \leq i \leq k$, $(c.a_1.a_2.\ldots.a_n \subseteq^* d_i) \in B$, $d_i \in T$ et $\bigwedge_{1 \leq i \leq k} d_i = \bot$.

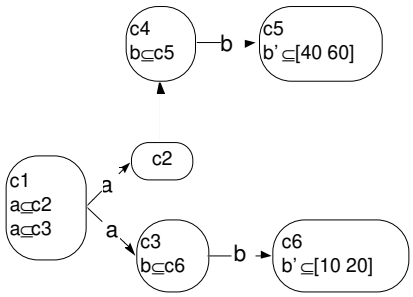


Figure 74 : l'intersection des domaines [40 60] et [10 20] est vide et la classe c_1 est donc incohérente.

démonstration.

si : soit un modèle élémentaire de la base (qui existe car la base est consistante) et $\hat{I}$ sa fonction d'interprétation. Une classe est incohérente si son interprétation est vide dans tous les modèles. Par l'absurde, supposons que $\hat{I}(c) \neq \emptyset$ et donc $o_c \in \hat{I}(c)$. On a $\hat{I}(a_n)(\hat{I}(a_{n-1})(\ldots \hat{I}(a_1)(\hat{I}(o_c))\ldots) \in \bigcap_i \hat{I}(d_i) = \hat{I}(\wedge_i d_i) = \emptyset$, ce qui est impossible donc la classe est incohérente.

seulement si : une classe est incohérente seulement si son interprétation est vide dans tous les modèles de la base; pour tout modèle :

$$I_B[c] \subseteq D \cap \bigcap_{c';\,(c \leq c') \in B} I_B[c'] \cap \bigcap_{a,d;\,(c.a \subseteq d) \in B} \{i \in D; I_B[a](i) \in I_B[d]\} = \emptyset$$

$$= \bigcap_{\substack{a,d;\,(c'.a \subseteq d) \in B \\ (c \leq^* c') \in B}} \{i \in D; I_B[a](i) \in I_B[d]\}$$

$$= \bigcap_{a,d;\,(c.a \subseteq^* d) \in B} \{i \in D; I_B[a](i) \in I_B[d]\}$$

Il existe donc un attribut particulier $a \in A$, tel que $\{i \in D; I_B[a](i) \in \bigcap_{d;(c.a \subseteq^* d) \in B} I_B[d]\} = \emptyset$, d'où

$$\bigcap_{d;(c.a \subseteq^* d) \in B} I_B[d] = \emptyset$$

si pour tout d, $d \in T$, le résultat est prouvé : $\bigwedge_{(c.a \subseteq^* d) \in B} d = \emptyset$.

s'il existe $d_1 \in T$ et $d_2 \in C$, $I_B[c] \subseteq I_B[d_1] \cap I_B[d_2] = \emptyset$.

si pour tout d, $d \in C$, le résultat est prouvé par récurrence sur le nombre d'attributs nécessaire pour obtenir un attribut qui est restreint par un domaine :

$$\bigcap_{(c.a \subseteq^* d) \in B} I_B[d] = \bigcap_{(c.a \subseteq^* d) \in B} \left(\bigcap_{(d.a' \subseteq^* d') \in B} \{i \in D; I_B[a'](i) \in I_B[d']\} \right)$$

$= \bigcap_{(c.a.a' \subseteq^* d') \in B} \{i \in D; I_B[a'](i) \in I_B[d']\}$ et donc il existe a' particulier $a' \in A$, tel que :

$$\bigcap_{(c.a.a' \subseteq^* d) \in B} I_B[d] = \emptyset$$

Cette récurrence est bien fondée vu que le nombre d'attributs et de classes est fini et que les références circulaires ne sont pas autorisées. ◊

5. Complétude faible

Les lemmes suivants seront utilisés pour démontrer la complétude faible. Ils donnent pour chaque type d'assertion, conséquence de la base, les assertions qui doivent appartenir à la base.

LEMME 9 :
Si la base B est consistante, si $B \models c \leq c'$ alors $(c \leq^* c') \in B$.

démonstration. Par l'absurde : Soit l'interprétation $M = \langle D', I' \rangle$, basée sur le modèle élémentaire $\langle D, \hat{I} \rangle$ (comme la base est consistante, $\langle D, \hat{I} \rangle$ est un modèle de la base), définie par :

- D' est augmenté de i tel que $i \notin D$,
- pour toutes les classes telles qu'il existe une chaîne de c à c'',

  $I'(c'') = \{I'(o) \text{ tel que } (o \in c'') \in B\} \cup \bigcup_{(c''' \leq c'') \in B} I'[c'''] \cup \{i\}$,
- les autres définitions n'étant pas modifiées.

Donc $\forall c''$ telle qu'il existe une chaîne de c à c'', $i \in I'(c'')$et $i \notin I'(c')$ car il n'existe pas de chaîne de c à c'. Donc $I'(c) \not\subset I'(c')$ et $M=<D',I'>$ est un modèle de B (car la preuve des différentes assertions n'est pas modifiée par ces changements de définitions par rapport au modèle élémentaire), donc $B \not\models c \leq c'$ d'où l'absurdité. ◊

LEMME 10 :

Si la base B est consistante et si $B \models c.a \subseteq d$ avec $d \in T$ alors

soit $d = \bigwedge_{(c.a \subseteq d_i) \in B} d_i \wedge \bigwedge_{\substack{B \models c \leq c_j \\ (c_j.a \subseteq d_{jk}) \in B}} d_{jk}$

soit d est un domaine non minimal, c'est-à-dire $\exists d' <:_t d \; B \models c.a \subseteq d'$

démonstration. Si d est minimal, c'est-à-dire si $\forall d' <:_t d \; B \not\models c.a \subseteq d'$, montrons alors que $d = \bigwedge_{(c.a \subseteq d_i) \in B} d_i \wedge \bigwedge_{\substack{B \models c \leq c_j \\ (c_j.a \subseteq d_{jk}) \in B}} d_{jk} = \Delta$.

• $B \models c.a \subseteq \Delta$ (car $B \models c.a \subseteq d_i$ pour tout $(c.a \subseteq d_i) \in B$ et $B \models c.a \subseteq d_{jk}$ pour tout $B \models c \leq c_j$ et $(c_j.a \subseteq d_{jk}) \in B$) donc $\neg \Delta <:_t d$.

• Δ est minimal car

$$\hat{I}(a|c) = \tau \cap \bigcap_{(c.a \subseteq d_i) \in B} \hat{I}[d_i] \cap \bigcap_{(c \leq c') \in B} \hat{I}[a|c']$$

$$= \tau \cap \bigcap_{(c.a \subseteq d_i) \in B} \hat{I}[d_i] \cap \bigcap_{\substack{B \models c \leq c' \\ (c_j.a \subseteq d_{jk}) \in B}} \hat{I}[d_{jk}] = \hat{I}(\Delta),$$

$\hat{I}(a|c) \not\subset \hat{I}(d')$ et $<D,\hat{I}>$ est modèle (car la base est consistante) donc $\not\models_{\hat{I}} c.a \subseteq d'$ pour $d' <:_t \Delta$.

• Or $\models_{<D,\hat{I}>} c.a \subseteq \Delta \wedge_t d$, donc $\neg \Delta \wedge_t d <:_t \Delta$ (d'après le second point), d'où $\Delta \wedge_t d =_t d$. De plus, $\neg \Delta <:_t d$ (d'après le premier point) donc $d = \Delta$. ◊

LEMME 11 :

Si la base B est consistante et si $B \models c.a \subseteq c'$ avec $c' \in C$ alors il existe c'' et c''' tels que $(c''.a \subseteq c''') \in B$, $(c \leq^* c'') \in B$ et $(c''' \leq^* c') \in B$.

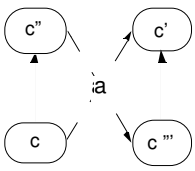


Figure 75 : condition du Lemme 11.

démonstration. Construisons un modèle $<D',I>$ basé sur le modèle élémentaire $<D,\hat{I}>$ et montrons que $\models_{<D,I>} c.a \subseteq c'$ seulement si la condition de la proposition est vérifiée.

$D'=D\cup\{o\}$ avec $o\notin D$.

$I(c'') = \hat{I}(c'')\cup\{o\}$ pour tout c''' tel que $(c\leq^* c'')\in B$ et $(c''.a\subseteq c''')\in B$.

La définition de I est identique à celle de $\hat{I}$ pour le reste.

Pour montrer que $<D',I>$ est un modèle, les seuls points à vérifier sont :

(1) $I(a|c)\subseteq I(a|c')$ si $I(c)\subseteq I(c')$

(2) $I(a|c)\subseteq I(c')$ si $(c.a\subseteq c')\in B$ pour les classes dont la définition a été modifiée par rapport à $\hat{I}$.

(1) $I(c)\subseteq I(c')$ si et seulement si $(c\leq^* c')\in B$ donc la démonstration est identique à celle du Lemme 7 (2).

(2) si $(c.a\subseteq c')\in B$, $\hat{I}(a|c)\subseteq\hat{I}(c')$; de plus, si pour toutes les classes c telles que $(c\leq^* c)\in B$ et $(c.a\subseteq c')\in B$, $o\in I(c')$, alors $I(a|c)=\hat{I}(a|c)\cup\{o\}$ et $I(c') =\hat{I}(c')\cup\{o\}$ d'où, $I(a|c)\subseteq\hat{I}(c')\cup\{o\}=I(c')$; sinon $I(a|c)=\hat{I}(a|c)\subseteq\hat{I}(c')\subseteq\hat{I}(c')\cup\{o\}=I(c')$.

Donc $<D',I>$ est un modèle.

$o\in I(a|c)$ car $I(a|c)=\bigcap_{\substack{(c''.a\subseteq c''')\in B\\(c\leq^* c'')\in B}} I(c''')$ et $o\in I(c'')$ pour c'' tel que $(c\leq^* c'')\in B$ et $(c''.a\subseteq c''')\in B$.

Comme $B\models c.a\subseteq c'$; $I(a|c)\subseteq I(c')$, donc $o\in I(c')$. Donc, il existe c'' tel que $(c''\leq^* c')\in B$ et $o\in I(c'')$ par définition donc $(c\leq^* c'')\in B$ et $(c''.a\subseteq c''')\in B$. ◊

LEMME 12 :

Si la base B est consistante et $B\models o.a=v$ alors $(o.a=v)\in B$ ou il existe $c_1,\ldots, c_n$ tels que $\forall i$ $B\models o\in c_i$, $B\models c_i.a\subseteq d_i$ et $\bigwedge_i\{d_i\} =\{v\}$ (avec $(\wedge d_i)$-$d_j\neq\{v\}$).

démonstration. Par l'absurde : Soit $B\models o.a=v$, $(o.a=v)\notin B$ et $\forall c_i$, $B\models o\in c_i$ et $B\models c_i.a\subseteq d_i$ et $\bigwedge_i\{d_i\}\neq\{v\}$. Il est alors possible de construire un modèle élémentaire tel que $o.a\in \bigwedge_i\{d_i\}-\{v\}$; le modèle élémentaire étant un modèle (car la base est consistante), cela a pour conséquence que $B\not\models o.a=v$ ce qui est contraire à l'hypothèse. ◊

LEMME 13 :

Si la base B est consistante et $B\models o.a=o'$ alors $(o.a=o')\in B$.

démonstration. Le modèle élémentaire satisfait : $\hat{I}(a)(\hat{I}(o))=K_B(a)(K_B(o))=K_B(o')$ si $(o.a=o')\in B$ et $\hat{I}(a)(\hat{I}(o))\in \bigcap_{\hat{I}(o)\in K_B(c)}\hat{I}(a|c)$ sinon. Comme $B\models o.a=o'$, $\hat{I}(a)(\hat{I}(o))=\hat{I}(o')$, ce qui est

vérifié dans le premier cas; dans le second cas, si $\bigcap_{\hat{I}(o)\in K_B(c)} \hat{I}(a|c)$ est un singleton, on peut construire un modèle de la base dans lequel l'interprétation des classes référencées par les classes d'appartenance de o contient une instance supplémentaire arbitraire, ce qui ne permet pas alors de déduire $B\models o.a=o'$. Donc, $(o.a=o')\in B$ est une condition nécessaire pour déduire $B\models o.a=o'$. ◊

PROPOSITION 7 : complétude faible

Soient une base B consistante et une assertion δ, si $B\models\delta$ alors $B\vdash\delta$.

démonstration. Ce théorème est démontré en 6 étapes qui correspondent à la démonstration pour chacun des 6 types d'assertions possibles d'une base. L'ordre de démonstration est basé sur la remarque suivante : les assertions du type $(c\leq c')$ ne peuvent être déduites d'aucune autre; les assertions du type $(c.a\subseteq d)$ et $(c.a\subseteq c')$ peuvent être déduites en utilisant des assertions du type $(c\leq c')$; par contre les assertions du type $(o\in c)$ et $(o.a=v)$ peuvent être déduites en utilisant des assertions des cinq types et en particulier, elles permettent de se déduire entre elles. Les assertions du type $(o.a=o')$ ne peuvent pas être déduites. Lorsque l'un des cas est démontré, il est utilisable dans la suite de la démonstration. De plus, comme la base est consistante, $<D,\hat{I}>$ est un modèle de la base.

$B\models c\leq c' \Rightarrow B\vdash c\leq c'$: Si $B\models c\leq c$, alors le résultat est acquis par la règle de (≤-réflexivité).

D'après le Lemme 9 si $B\models c\leq c'$, alors $(c\leq^* c')\in B$ et donc d'après la Proposition 4 , $B\vdash c\leq c'$.

$B\models c.a\subseteq d \Rightarrow B\vdash c.a\subseteq d$: d'après le Lemme 10, d est non minimal (2) ou

$$d = \bigwedge_{(c.a\subseteq d_i)\in B} d_i \wedge \bigwedge_{\substack{B\models c\leq c_j \\ (c_j.a\subseteq d_{jk})\in B}} d_{jk} \quad (1).$$

(1) si d est minimal, $B\models c.a\subseteq \bigwedge_{(c.a\subseteq d_i)\in B} d_i \wedge \bigwedge_{\substack{B\models c\leq c_j \\ (c_j.a\subseteq d_{jk})\in B}} d_{jk}$; pour tout $B\models c\leq c_j$ et $(c_j.a\subseteq d_{jk})\in B$, $B\vdash c\leq c_j$ (démontré ci-dessus) et $B\vdash c_j.a\subseteq d_{jk}$ et donc en utilisant la règle de (d-héritage), $B\vdash c.a\subseteq d_{jk}$. De plus, pour tout $(c.a\subseteq d_i)\in B$, $B\vdash c.a\subseteq d_i$ et donc en utilisant la règle de (d-intersection) sur toutes les assertions du type $c.a\subseteq d$ pour c et a donnés et d quelconque :

$$B\vdash c.a\subseteq \bigwedge_{(c.a\subseteq d_i)\in B} d_i \wedge \bigwedge_{\substack{B\models c\leq c_j \\ (c_j.a\subseteq d_{jk})\in B}} d_{jk},$$

d'où $B \vdash c.a \subseteq d$

(2) Si d est tel que $B \models c.a \subseteq d$ et $\exists d' <:_t d$ tel que $B \models c.a \subseteq d'$, il existe (au moins) un $d'' <:_t d'$ «minimal» (c'est-à-dire tel que $B \models c.a \subseteq d''$ et $\forall d' <:_t d''$ $B \not\models c.a \subseteq d'$) et donc $B \vdash c.a \subseteq d''$ d'après ce qui précède et par la règle de (d-clôture) $B \vdash c.a \subseteq d$.

$B \models c.a \subseteq c' \Rightarrow B \vdash c.a \subseteq c'$: d'après le Lemme 11, il existe c''' et c'' tels que $(c'''.a \subseteq c'') \in B$, $(c \leq^* c''') \in B$ et $(c'' \leq^* c') \in B$. D'où $B \vdash c \leq c'''$ et $B \vdash c'' \leq c'$ et donc par les règles (d-héritage) et (d-clôture), $B \vdash c.a \subseteq c'$.

$B \models o \in c \Rightarrow B \vdash o \in c$: $\hat{I}(o) \in \hat{I}(c)$, donc $\hat{I}(o) \in K_B(c')$ avec $(c' \leq^* c) \in B$ c'est-à-dire qu'il existe $c' \in C$ tel que $\hat{I}(o) \in \mathrm{OA}(c')$ Donc, d'après la définition de $\mathrm{OA}(c')$, soit o est tel que $(o \in c') \in B$, et dans ce cas $B \vdash o \in c$ par la règle de ($\in$-clôture), soit il existe $a_1, a_2, \ldots, a_n \in A$, $n \geq 1$, $o' \in O$, $c'' \in C$ tels que $(o' \in c'') \in B$, $(o'.a_1.a_2.\ldots.a_n = {}^*o) \in B$ et $(c''.a_1.a_2.\ldots.a_n \subseteq {}^*c') \in B$; d'après la Proposition 16, $B \vdash o \in c'$ et par application de la (c-clôture), $B \vdash o \in c$.

$B \models o.a = v \Rightarrow B \vdash o.a = v$: Si $(o.a = v) \in B$ le résultat est immédiat. Si $(o.a = v) \notin B$ alors d'après le Lemme 12, il existe $c_1, \ldots, c_n$ tels que $\forall i$ $B \models o \in c_i$ et $B \models c_i.a \subseteq d_i$ et $\wedge_i d_i = \{v\}$.

Par conséquent, $\forall i$ $B \vdash o \in c_i$ et $B \vdash c_i.a \subseteq d_i$, d'après ce qui a été démontré précédemment et en appliquant la règle de (o-valuation), $B \vdash o.a = v$.

$B \models o.a = o' \Rightarrow B \vdash o.a = o'$: Si $B \models o.a = o'$, alors $(o.a = o') \in B$ d'après le Lemme 13 et $B \vdash o.a = o'$.

Conclusion : pour chacun des six types d'assertion, le théorème est démontré. ◊

6. Forme première

Les propriétés de la forme première sont démontrées ci-dessous.

PROPOSITION (8 ET) 257 : FORME PREMIÈRE

Soit une base B. La forme première de B se calcule par :

$\mathrm{FP}(B) = \mathrm{FN}(B)$

– $\{(c.a \subseteq d)\ ;\ \{c_i.a \subseteq d_i \quad c \leq c_i\} \in \mathrm{Cn}'(B),\ \bigwedge_i d_i = d\}$ (d-anti-intersection)

– $\{(o.a = v)\ ;\ \{c_i.a \subseteq d_i \quad o \in c_i\} \in \mathrm{Cn}'(B)$ et $\{v\} = \bigwedge_i d_i\}$ (o-anti-valuation)

– $\{(c.a \subseteq c')\ ;\ (c''.a \subseteq c'),\ (c \leq c'') \in \mathrm{Cn}'(B)\}$ (c-anti-héritage)

– $\{(o \in c)\ ;\ (o' \in c'),\ (c'.a \subseteq c'),\ (o'.a = o) \in \mathrm{Cn}'(B)\}$ ($\in$-anti-inférence)

démonstration.La démonstration se décompose en trois points :

(1) $B \vdash \mathrm{FP}(B)$

(2) $\mathrm{FP}(B) \vdash B$

(3) Pour tout assertion δ appartenant à FP(B), $\mathrm{FP}(B) - \{\delta\} \nvdash \delta$.

(1) $B \vdash \mathrm{Cn}(B)$ et $\mathrm{FP}(B) \subseteq \mathrm{Cn}(B)$.

(2) Chaque type d'assertion de B est étudié pour montrer qu'une telle assertion peut être déduite à partir de FP(B); l'ordre de traitement des assertions permet d'utiliser les points précédemment démontrés :

$c \leq c'$: si $(c \leq c') \in \mathrm{Cn}(B) - \mathrm{FP}(B)$ alors il a été supprimé par la règle de ($\leq$-réduction), c'est-à-dire qu'il existe une classe intermédiaire (($c \leq c''$) et ($c'' \leq c'$) $\in \mathrm{Cn}(B)$); en procédant par récurrence, cette assertion est retirée dans la forme première s'il existe $(c \leq^* c') \in B$ et toutes les assertions appartiennent à la forme première (car il n'est pas possible de les supprimer de la clôture déductive). L'application de la règle de ($\leq$-transitivité) (sur toute la chaîne), donne alors $\mathrm{FP}(B) \vdash c \leq c'$.

$c.a \subseteq d$: le résultat est prouvé par récurrence sur la distance de la classe à la racine. Pour la classe racine, l'assertion a été retirée par la règle de (d-normalisation); il existe $(c.a \subseteq d') \in \mathrm{Cn}'(B)$ tel que $d' <:_t d$; soit l'assertion avec le domaine minimal (unique car dans $\mathrm{Cn}'(B)$, il y a toutes les intersection de domaines); cette assertion appartient à FP(B) car elle ne peut pas être retirée de Cn(B) car la (d-normalisation) ne s'appliquent pas et à partir d'elle, toutes les assertions de restriction de domaine peuvent être redéduites.

Soit à présent le cas d'une classe quelconque de la hiérarchie avec comme hypothèse de récurrence que pour toutes ses super-classes $\mathrm{FP}(B) \vdash c'.a \subseteq d$ si $c'.a \subseteq d \in \mathrm{Cn}(B)$. Cette assertion a pu être retirée par deux règles différentes :
(d-normalisation) : s'il existe $(c.a \subseteq d') \in \mathrm{Cn}'(B)$ tel que $d' <:_t d$, on choisit l'assertion avec le domaine minimal et celle-ci appartient à F ou a été supprimée par (d-anti-intersection); pour ce dernier cas, le résultat est obtenu en utilisant l'hypothèse de récurrence.

(d-anti-intersection) : il existe un ensemble $\{c_i.a \subseteq d_i \quad c \leq c_i\} \in \mathrm{Cn'}(B)$ avec $\underset{i}{\wedge} d_i = d$.

En utilisant l'hypothèse de récurrence et la règle de (d-intersection) le nombre de fois nécessaires, $\mathrm{FP}(B) \vdash c.a \subseteq d$.

Le résultat est donc montré par récurrence.

$c.a \subseteq c'$: $(c.a \subseteq c') \in \mathrm{Cn}(B) - \mathrm{FP}(B)$, l'assertion a été retirée par la (c-réduction) ou (c-anti-héritage). Considérons l'assertion $(c''.a \subseteq c''') \in \mathrm{FP}(B)$ telle que $(c'' \leq c) \in \mathrm{Cn}(B)$, $(c' \leq c''') \in \mathrm{Cn}(B)$; une telle assertion existe sinon il n'est pas possible de supprimer $(c.a \subseteq c')$ et elle appartient à la forme première car elle ne peut être supprimée (on choisit celle pour laquelle il n'est pas possible de trouver des assertions pour la supprimer - vu qu'il n'y a pas de référence circulaire, ceci est valide). $\mathrm{FP}(B) \vdash c''.a \subseteq c'''$, $\mathrm{FP}(B) \vdash c'' \leq c$ et $\mathrm{FP}(B) \vdash c' \leq c''$ en utilisant l'un des points précédents de la démonstration. Donc en appliquant les règle de (c-clôture) et de (c-héritage), $\mathrm{FP}(B) \vdash c.a \subseteq c'$.

$o.a = o'$: il n'existe pas d'assertion telle que $(o.a = o') \in \mathrm{Cn}(B) - \mathrm{FP}(B)$, car aucune règle ne permet de supprimer ce type d'assertion de la clôture déductive (de la même manière qu'elle ne peut être déduite par le système déductif).

$o \in c$: si $(o \in c) \in \mathrm{Cn}(B) - \mathrm{FP}(B)$ alors il a été supprimé par la règle de ($\in$-réduction) ou ($\in$-anti-inférence), c'est-à-dire qu'il existe c' telle que $(c \leq c') \in \mathrm{Cn}(B)$ et $(o \in c') \in \mathrm{Cn}(B)$ ou il existe o', c', a tels que $(o' \in c') \in \mathrm{Cn}(B)$, $(c'.a \subseteq c) \in \mathrm{Cn}(B)$ et $(o'.a = o) \in \mathrm{Cn}(B)$.
Dans le premier cas : $\mathrm{FP}(B) \vdash c \leq c'$ et en choisissant c' la plus haute classe telle que $(o \in c') \in \mathrm{Cn'}(B)$, $(o \in c') \in \mathrm{FP}(B)$ (car cette assertion n'a pas pu être retirée de la forme première). Donc par application de la règle de ($\in$-clôture), $\mathrm{FP}(B) \vdash o \in c$.

Dans le second cas : $\mathrm{FP}(B) \vdash (c'.a \subseteq c)$, $\mathrm{FP}(B) \vdash (o'.a = o)$. De plus, comme il n'y a pas de référence circulaire, il est possible de trier les assertions $(o' \in c')$ avec en premier celles qui ne sont référencées par aucune instance et pour lesquelles seul le premier cas s'applique; ensuite par récurrence sur le nombre d'assertion du type $(o'.a = o)$ qui relie une instance à une instance non référencée, on démontre que $\mathrm{FP}(B) \vdash (o' \in c')$. D'où, en appliquant la règle de ($\in$-inférence), $\mathrm{FP}(B) \vdash o \in c$.

$o.a = v$: si $(o.a = v) \in \mathrm{Cn}(B) - \mathrm{FP}(B)$, alors $(c_i.a \subseteq d_i) \in \mathrm{Cn'}(B)$ et $(o \in c_i) \in \mathrm{Cn'}(B)$ pour un ensemble de i tels que $\{v\} = \underset{i}{\wedge} d_i$. Des points précédents de la démonstration, $\mathrm{FP}(B)$ $c_i.a \subseteq d_i$ et $\mathrm{FP}(B) o \in c_i$ et donc $\mathrm{FP}(B) o.a = v$ par la règle de (o-valuation).

Le résultat est donc montré pour toutes les assertions de la base.

(3) FP(B)–{δ} sera noté F pour simplifier l'écriture.

Pour le type d'assertion δ, on montre par l'absurde que si l'assertion δ est déductible de F alors δ n'appartient pas à la forme première car il existe une règle permettant de l'oter.

$c \leq c'$: si $F \vdash c \leq c'$, d'après la Proposition 4 , $(c \leq^* c') \in F$. Donc, comme $(c \leq^* c) \notin F$ (par définition de la forme première) et $(c \leq c') \notin F$ (car cette assertion a été explicitement otée de la forme première), il existe c'' distincte de c et de c' telle que $(c \leq c'') \in \mathrm{Cn}(F)$ et $(c'' \leq c') \in \mathrm{Cn}(F)$, d'où, comme $\mathrm{Cn}(F) \subseteq \mathrm{Cn}(B)$, par application de la ($\leq$-réduction), $(c \leq c') \notin \mathrm{FP}(B)$.

$o \in c$: si $F \vdash o \in c$, il existe $c' \in C$ tel que $(o \in c') \in \mathrm{Cn}(F)$ et $(c' \leq c) \in \mathrm{Cn}(F)$ ou il existe $a \in A$, $o' \in O$ et $c' \in C$ tels que $(o' \in c') \in \mathrm{Cn}(F)$, $(c'.a \subseteq c) \in \mathrm{Cn}(F)$ et $(o.a = o') \in \mathrm{Cn}(F)$. Comme $\mathrm{Cn}(F) \subseteq \mathrm{Cn}(B)$, par application de ($\in$-réduction) ou ($\in$-anti-inférence), $(o \in c) \notin \mathrm{FP}(B)$.

$c.a \subseteq d$: si $F \vdash c.a \subseteq d$ alors $F \models c.a \subseteq d$ et donc d'après le Lemme 10, soit d est un domaine non minimal, soit $d = \bigwedge_{(c.a \subseteq d_i) \in F} d_i \wedge \bigwedge_{\substack{F \models c \leq c_j \\ (c_j.a \subseteq d_{jk}) \in F}} d_{jk}$.

Si d est un domaine non minimal, il existe $F \vdash c.a \subseteq d'$ avec $d' <:_t d$ et dans ce cas, $\delta \notin \mathrm{FP}(B)$ d'après la règle de (d-normalisation).

Si $d = \bigwedge_{(c.a \subseteq d_i) \in F} d_i \wedge \bigwedge_{\substack{F \models c \leq c_j \\ (c_j.a \subseteq d_{jk}) \in F}} d_{jk}$, $d = \bigwedge_{\substack{(c \leq c_j) \in Cn'(B) \\ (c_j.a \subseteq d_{jk}) \in Cn'(B)}} d_{jk}$ car il n'existe pas d'assertion $c.a \subseteq d$ dans F (vu que dans la forme première, il n'existe qu'une seule assertion de ce type pour c et a donnés et que cette assertion dans FP(B) est δ). Dans ce cas, par la règle de (d-anti-intersection), $\delta \notin \mathrm{FP}(B)$.

$c.a \subseteq c'$: si $F \vdash c.a \subseteq c'$ alors $F \models c.a \subseteq c'$ et donc d'après le Lemme 11, il existe c''' et c'' tels que $(c'''.a \subseteq c'') \in F$, $(c \leq^* c''') \in F$ et $(c'' \leq^* c') \in F$. D'où, $(c \leq c''') \in \mathrm{Cn}(F)$ et $(c'' \leq c') \in \mathrm{Cn}(F)$. Comme $\mathrm{Cn}(F) \subseteq \mathrm{Cn}(B)$, par application de (c-réduction), $(c.a \subseteq c') \notin \mathrm{FP}(B)$

$o.a = v$: si $F \vdash o.a = v$, $F \models o.a = v$ et donc d'après le Lemme 12, il existe des classes $c_1, \ldots, c_n$, telles que $\forall i \; F \models o \in c_i$ et $F \models c_i.a \subseteq d_i$ et $\wedge_i \{d_i\} = \{v\}$ et comme $F \subseteq \mathrm{Cn}'(B)$, par application de la (o-anti-valuation), $(o.a = v) \notin \mathrm{FP}(B)$.

$o.a=o'$: si $F\vdash o.a=o'$, $F\models o.a=o'$ et donc d'après le Lemme 12, il existe des classes $c_1,\ldots,$ c_n, telles que $\forall i\ F\models o\in c_i$ et $F\models c_i.a\subseteq d_i$ et $\bigwedge_i\{d_i\}=\{o'\}$ et comme $F\subseteq \mathrm{Cn}'(B)$, par application de la (o-anti-valuation), $(o.a=o')\notin \mathrm{FP}(B)$. ◊

PROPOSITION 9 : UNICITÉ DE LA FORME PREMIÈRE

Soit une base B ; il existe une unique forme première FP(B).

démonstration.Raisonnons par l'absurde. Soit une base B. Supposons qu'il existe une base FP' satisfaisant les conditions de forme première :

(1) $B \vdash$ FP'

(2) FP' $\vdash B$

(3) Pour tout assertion δ appartenant à FP', FP'$-\{\delta\} \nvdash \delta$.

Par transitivité de $\vdash$ et en utilisant (1), (2) et les propriétés similaires pour FP(B), on a : FP'$\vdash$FP(B) et FP(B)$\vdash$FP'. De plus, FP(B)$\subseteq$FP'. En effet, soit une assertion δ appartenant à FP(B) ; comme FP'$\vdash$FP(B), il existe un ensemble d'assertions de FP' et donc de Cn(B) permettant de déduire δ ; or, d'après la caractérisation de FP(B), si un tel ensemble d'assertions ne contenant pas δ existe, δ n'appartient pas à la forme première. Donc $\delta\in$ FP', donc FP(B)$\subseteq$FP'. Comme FP' est minimal (3), FP'=FP(B). ◊

PROPOSITION 10

La taille maximale de la forme première est de l'ordre de :

$(|O|+|C|)*(|A|+|C|)$ dans le cas général

$(|O|*|A|)+|C|*(|A|+|C|)$ sans multi-instanciation

$|O|*(|A|+|C|)+(|A|*|C|)$ sans multi-spécialisation

$(|O|+|C|)*|A|$ sans aucun des deux

démonstration. La forme première ne préserve plus qu'une valeur par attribut d'instance ($|T_O|*|T_A|$), au maximum un domaine par attribut de classe ($|T_C|*|T_A|$) (en effet, la clôture contient toutes les intersections possibles de domaines pour *c.a* donné, donc en particulier l'intersection de toutes les restrictions de domaine, qui est contenu dans tous les autres domaines). Par ailleurs, toute classe peut être attachée à toute autre ($|T_C|^2$, même si l'on considère qu'il ne reste que la réduction transitive de l'ordre); cependant, si l'on se restreint à la mono-spécialisation, cette valeur tombe à $|T_C|$. Toute instance peut être attachée à toute classe ($|T_O|*|T_C|$), mais en se restreignant à la mono-instanciation, la valeur devient $|T_O|$. ◊

PROPOSITION 11

Cn(FP(B))=Cn(B)

FP(Cn(B))=FP(B)
FP(FP(B)) = FP(B) (stabilité)

démonstration. Les quatre premières sont immédiates d'après la définition 12. Pour la cinquième : on a Cn(FP(B))=Cn(B) et l'application de FP à ces deux termes va en enlever les mêmes éléments. ◊

PROPOSITION 12
$\forall B, B'$, les trois conditions suivantes sont équivalentes :
(1) FP(B')$\vdash$FP(B)
(2) $B'\vdash B$
(3) Cn(B)$\subseteq$Cn(B')

démonstration.

(1) $\Rightarrow$ (2) : $B'\vdash$FP(B'), FP(B')$\vdash$FP(B) et FP(B) $\vdash B$, d'où $B'\vdash B$.

(2) $\Rightarrow$ (3) : $B\subseteq$Cn(B') car $B'\vdash B$ donc Cn(B)$\subseteq$Cn(Cn(B'))=Cn(B').

(3) $\Rightarrow$ (1) : Cn(B)$\subseteq$Cn(B'), FP(B')$\vdash$Cn(B') donc FP(B')$\vdash$Cn(B) et FP(B')$\vdash$FP(B). ◊

PROPOSITION 13
$\forall B, B'$, les trois conditions suivantes sont équivalentes :
(1) FP(B) = FP(B')
(2) $B\vdash B'$ et $B'\vdash B$
(3) Cn(B) = Cn(B')

démonstration.

(2) $\Leftrightarrow$ (3) : immédiat d'après la proposition précédente.

(3) $\Rightarrow$ (1) : immédiat vu la définition de la forme première.

(1) $\Rightarrow$ (3) : immédiat d'après la proposition précédente. ◊

7. Base révisée

Enfin, les propriétés concernant la révision, présentée au chapitre III sont démontrées. En fait, les propriétés portent princiaplement sur les bases contractées.

PROPOSITION 21 : CARACTÉRISATION SÉMANTIQUE DE LA MINIMALITÉ

B' est une base contractée minimale de (B,δ) ssi

(1) $\forall\gamma$ tel que $B\models\gamma$ et $B'\cup\{\delta\}\not\models\gamma$ alors $B'\cup\{\delta\}\cup\{\gamma\}$ est inacceptable.

démonstration.

seulement si) par l'absurde, B' est minimale et il existe γ tel que $B \models \gamma$ et $B' \cup \{\delta\} \not\models \gamma$ et $B' \cup \{\delta\} \cup \{\gamma\}$ est acceptable. Soit $B'' = B' \cup \{\gamma\}$; B'' est une base contractée car $B'' \cup \{\delta\}$ est acceptable et $Cn(B'') \subseteq Cn(B)$. De plus, $B'' \models B'$ et $B' \not\models B''$, donc $B'' \propto B'$, donc B' n'est pas minimale, d'où la contradiction.

si) par l'absurde, B' est une base contractée de (B,δ) satisfaisant (1) et il existe $B'' \neq B'$ telle que B'' est une base contractée de (B,δ) et $B'' \propto B'$. Comme $Cn(B') \subset Cn(B'')$, il existe γ tel que $B'' \models \gamma$ et $B' \cup \{\delta\} \not\models \gamma$ alors $B' \cup \{\delta\} \cup \{\gamma\}$ est inacceptable et comme $B' \cup \{\delta\} \cup \{\gamma\} \subset Cn(B'')$, B'' est inacceptable d'où la contradiction. Si $\forall \gamma$ tel que $B'' \models \gamma$ alors $B' \cup \{\delta\} \models \gamma$, alors $B' \cup \{\delta\} \models B'' \cup \{\delta\}$ et $B'' \cup \{\delta\} \models B' \cup \{\delta\}$ et comme B' et B'' sont exprimées sous forme première $B' = B''$, il y a contradiction. ◊

PROPOSITION 22 : CARACTÉRISATION SYNTAXIQUE DE LA MINIMALITÉ

Si B' et B'' sont deux bases contractées de (B,δ), B' est plus petite que B'' ssi $B'_- \subseteq B''_-$ et $B' \vdash B''_+$.

démonstration.

si) $B'_- \subseteq B''_- \Rightarrow FP(B) - B''_- \subseteq FP(B) - B'_-$

$\Rightarrow FP(B) - B''_- \subseteq FP(B) - B'_- \cup B'_+$

$\Rightarrow FP(B) - B''_- \subseteq Cn(FP(B) - B'_- \cup B'_+) = Cn(B')$

De plus, $B' \vdash B''_+$ donc $B''_+ \subseteq Cn(B')$

Donc, $FP(B) - B''_- \cup B''_+ \subseteq Cn(B')$ ce qui équivaut à $Cn(B'') \subseteq Cn(B')$.

seulement si)

• $Cn(B'') \subseteq Cn(B') \Rightarrow B''_+ \subseteq Cn(B')$ car $B''_+ \subseteq B'' \subseteq Cn(B'')$.

• Il faut montrer par l'absurde que $B'_- \subseteq B''_-$: soit δ tel que $\delta \in B'_-$ et $\delta \notin B''_-$. Vu que $\delta \notin B''_-$ et $\delta \in FP(B)$ (par définition), $\delta \in Cn(B'')$. De plus, B' est sous forme première donc d'après la minimalité de la forme première (Proposition 9), $B' \not\vdash \delta$. Ainsi, $\delta \in Cn(B'')$ et $\delta \notin Cn(B')$, d'où la contradiction et donc $B'_- \subseteq B''_-$. ◊

PROPOSITION 23 : INTERPRÉTATION "OBJET" DE LA MINIMALITÉ

B' est une base contractée minimale de (B,δ) ssi :

(0) B' est une base contractée de (B,δ).

(1) $\forall B''$ base contractée de (B,δ), si $B''_- \subseteq B'_-$ et $B'_+ \subseteq B''_+$, alors $B'=B''$.

(2) $\forall (c \leq c') \in B'_+$, $\forall c_1 \neq c'$, si $B' \vdash c_1 \leq c'$ et $(c \leq c_1) \in \mathrm{Cn}(B)$, alors $(c \leq c_1)$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.

(3) $\forall (o \in c) \in B'_+$, $\forall c' \neq c$, si $B' \vdash c' \leq c$ et $(o \in c') \in \mathrm{Cn}(B)$, alors $(o \in c')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.

(4) $\forall$ $(c.a \subseteq d) \in B'_+$, $\forall (c',d') \neq (c,d)$, si $d' \leq: d$, $B' \vdash c \leq c'$ et $(c'.a \subseteq d') \in \mathrm{Cn}(B)$, alors $(c'.a \subseteq d')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.

(4') $\forall$ $(c.a \subseteq d) \in B'_+$, $\forall (c',d') \neq (c,d)$, $d,d' \in C$, si $B' \vdash d' \leq d$, $B' \vdash c \leq c'$ et $(c'.a \subseteq d') \in \mathrm{Cn}(B)$, alors $(c'.a \subseteq d')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.

(5) $\forall (o \in c) \in B'_+$, $\forall c' \neq c$, o', a si $(c'.a \subseteq c) \in \mathrm{Cn}(B)$, $(o' \in c') \in \mathrm{Cn}(B)$ et $(o'.a=o) \in \mathrm{Cn}(B)$, alors l'une de ces trois assertions n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.

(6) $\forall (o.a=v) \in B'_+$, pour tout ensemble d'assertions $\{c_i.a \subseteq d_{ij}, o \in c_i\} \in \mathrm{Cn}(B)$ minimal tel que $\{v\} = \wedge d_{ij}$, alors l'une des assertions n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.

Les propositions suivantes seront utiles pour la démonstration de la partie «si» du théorème.

LEMME 14 :
Soit B' une base contractée de (B,δ) vérifiant les conditions (0), (1) et (2); s'il existe c, c' tels que $B \vdash c \leq c'$ et $B' \nvdash c \leq c'$, alors $(c \leq c')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$.

démonstration.

Il existe deux cas distincts selon l'existence d'une assertion du type $(c \leq c'')$ dans B'_+.
S'il existe une classe c'' telle que $(c \leq c'') \in B'_+$, alors :

- si $\mathrm{FP}(B) - B'_- \vdash c' \leq c''$ $(c' \neq c'')$, alors $(c \leq c'')$ n'est pas une assertion acceptable dans $B' \cup \{\delta\}$ d'après la condition (2) et donc B' n'est pas une base contractée de (B,δ).
- si $\mathrm{FP}(B) - B'_- \vdash c'' \leq c'$ $(c' \neq c'')$, alors $B' \vdash c \leq c'$ (par application de la règle de $\leq$-transitivité), ce qui est contraire aux hypothèses.

si $FP(B)-B'_{-} \nvdash c'\leq c''$ et $FP(B)-B'_{-} \nvdash c''\leq c'$ ($c'\neq c''$), alors considérons la base définie par $B''_{-}=B'_{-}$ et $B''_{+}=B'_{+}\cup\{c\leq c'\}$. Si ($c\leq c'$) est une assertion acceptable dans $B'\cup\{\delta\}$, alors B'' est une base contractée car B'' est exprimée sous forme première (car il n'existe pas de lien entre les deux classes c' et c'' et donc les assertions ($c\leq c''$) et ($c\leq c'$) peuvent appartenir toutes deux à la forme première) et il y a contradiction avec la condition (1) donc ($c\leq c'$) n'est pas une assertion acceptable dans $B'\cup\{\delta\}$.

S'il n'existe pas de classe c'' telle que $(c\leq c'')\in B'_{+}$, soit la base définie par $B''_{-}=B'_{-}$ et $B''_{+}=B'_{+}\cup\{c\leq c'\}$; si ($c\leq c'$) est une assertion acceptable dans $B'\cup\{\delta\}$, alors B'' est une base contractée car B'' est exprimée sous forme première et dans ce cas, il y a contradiction avec la condition (1). ◊

LEMME 15

Soit B' une base contractée de (B,δ) vérifiant les conditions (1) et (4); s'il existe c, a et d tels que $B\vdash c.a\subseteq d$ et $B'\nvdash c.a\subseteq d$, alors ($c.a\subseteq d$) n'est pas une assertion acceptable dans $B'\cup\{\delta\}$.

démonstration.

La démonstration se décompose en deux points selon que dans les sous-classes il existe des restrictions de domaine déductibles de ($c.a\subseteq d$) ou non.

S'il existe une classe c' telle que $B'\vdash c'\leq c$ ($c'\neq c$), $(c'.a\subseteq d')\in B'$ et $d\leq: d'$ alors $(c'.a\subseteq d')\in B'_{+}$ (car $(c'.a\subseteq d')\notin FP(B)$, car $d\leq: d'$ et $(c'.a\subseteq d)\in FP(B)$ car $B\vdash c.a\subseteq d$ et $B\vdash c'\leq c$) et en utilisant la condition (4), ($c.a\subseteq d$) n'est pas une assertion acceptable dans $B'\cup\{\delta\}$.

Si pour toute classe c' telle que $B'\vdash c'\leq c$ ($c'\neq c$), $(c'.a\subseteq d')\notin B'$ ou $(c'.a\subseteq d')\in B'$ et $\neg d\leq: d'$.

si $(c.a\subseteq d')\in B'_{+}$:

si $d'\leq: d$ alors $B'\vdash c.a\subseteq d$, ce qui est contraire aux hypothèses;

si $d<: d'$ alors ($c.a\subseteq d$) non acceptable d'après la condition (4);

si d' et d sont non comparables, alors la base définie par $B''_{-}=B'_{-}$ et $B''_{+}=B'_{+}\cup\{c.a\subseteq d\}$ est sous forme première et est donc une base contractée si ($c.a\subseteq d$) est acceptable; or

dans ce cas, d'après la condition (1), $B'=B''$, ce qui est contradictoire avec les hypothèses.

s'il n'existe pas d'assertion $(c.a\subseteq d')\in B'_+$, la base définie par $B''_- = B'_-$ et $B''_+ = B'_+ \cup \{c.a\subseteq d\}$ est sous forme première (car les assertions de restriction de domaine dans les sous-classes peuvent toujours appartenir à la forme première) et est donc une base contractée si $(c.a\subseteq d)$ est acceptable; or dans ce cas, d'après la condition (1), $B'=B''$, ce qui est contradictoire avec les hypothèses (car $B' \nvdash c.a\subseteq d$). ◊

LEMME 16

Soit B' une base contractée de (B,δ) vérifiant les conditions (1) et (4'); s'il existe c, a et $d\in C$ tels que $B\vdash c.a\subseteq d$ et $B' \nvdash c.a\subseteq d$, alors $(c.a\subseteq d)$ n'est pas une assertion acceptable dans $B'\cup\{\delta\}$.

démonstration.

La démonstration se décompose en deux points selon que dans les sous-classes il existe des restrictions de domaine déductibles de $(c.a\subseteq d)$ ou non.

S'il existe une classe c' telle que $B' \vdash c'\leq c$ $(c'\neq c)$, $(c'.a\subseteq d')\in B'$ et $B' \vdash d\leq d'$ alors $(c'.a\subseteq d')\in B'_+$ (car $(c'.a\subseteq d')\notin \mathrm{FP}(B)$, vu que la (c-réduction) s'applique car $B\vdash d\leq d'$ et $(c'.a\subseteq d)\in \mathrm{Cn}(B)$ car $B\vdash c.a\subseteq d$ et $B\vdash c'\leq c$) et en utilisant la condition (4'), $(c.a\subseteq d)$ n'est pas une assertion acceptable dans $B'\cup\{\delta\}$.

Si pour toute classe c' telle que $B' \vdash c'\leq c$ $(c'\neq c)$, $(c'.a\subseteq d')\notin B'$ ou $(c'.a\subseteq d')\in B'$ et $B' \nvdash d\leq d'$.

si $(c.a\subseteq d')\in B'_+$:

si $B'\vdash d'\leq d$ alors $B'\vdash c.a\subseteq d$, ce qui est contraire aux hypothèses;

si $B'\vdash d\leq d'$ alors $(c.a\subseteq d)$ n'est pas acceptable d'après la condition (4');

si d' et d sont non comparables, alors la base définie par $B''_- = B'_-$ et $B''_+ = B'_+ \cup \{c.a\subseteq d\}$

est sous forme première et est donc une base contractée si (c.a⊆d) est acceptable; or dans ce cas, d'après la condition (1), $B'=B''$, ce qui est contradictoire avec les hypothèses.

s'il n'existe pas d'assertion $(c.a\subseteq d')\in B'_+$, la base définie par $B''_- = B'_-$ et $B''_+ = B'_+ \cup \{c.a\subseteq d\}$ est sous forme première (car les assertions de restriction de domaine dans les sous-classes peuvent toujours appartenir à la forme première) et est donc une base contractée si $(c.a\subseteq d)$ est

acceptable; or dans ce cas, d'après la condition (1), $B'=B''$, ce qui est contradictoire avec les hypothèses (car $B' \nvdash c.a \subseteq d$). ◊

démonstration du théorème.

La première partie de la démonstration est relativement immédiate; la seconde nécessite plus de moyens.

seulement si) B' est une base contractée minimale de (B,δ).
(0) est immédiat.

(1) Si B'' est une base contractée de (B,δ), $B''_- \subseteq B'_-$ et $B'_+ \subseteq B''_+$, alors $B'' \vdash B'_+$, donc $B'' \propto B'$, et comme B' est une base minimale, $B''=B'$.

(2) Par l'absurde, supposons que $(c \leq c_1)$ soit acceptable dans $B' \cup \{\delta\}$ avec $B' \vdash c_1 \leq c'$, $(c \leq c') \in B'_+$ et $c_1 \neq c'$. La base B'' telle que $B''_- = B'_-$ et $B''_+ = B'_+ \cup \{c \leq c_1\} - \{c \leq c'\}$ est telle que $B'' \cup \{\delta\}$ est acceptable et B'' est donc une base contractée; de plus, $B'' \vdash B'_+$ (car $B'' \vdash c \leq c'$, vu que $(c \leq c_1) \in B''_+$ et $B'' \vdash c_1 \leq c'$) et $B'' \neq B'$ car $c_1 \neq c'$, donc B' n'est pas minimale, ce qui est contradictoire avec l'hypothèse.

(3) Par l'absurde, supposons que $(o \in c')$ soit acceptable dans $B' \cup \{\delta\}$ avec $B' \vdash c' \leq c$, $(o \in c) \in B'_+$ et $c \neq c'$. La base B'' telle que $B''_- = B'_-$ et $B''_+ = B'_+ \cup \{o \in c'\} - \{o \in c\}$ est sous forme première; $B'' \cup \{\delta\}$ est acceptable et B'' est donc une base contractée; de plus, $B'' \vdash B'_+$ (car $B'' \vdash o \in c$, vu que $(o \in c') \in B''_+$ et $B'' \vdash c' \leq c$) et $B'' \neq B'$ car $c \neq c'$, donc B' n'est pas minimale, ce qui est contradictoire avec l'hypothèse.

(4) Par l'absurde, si $(c'.a \subseteq d')$ est acceptable dans $B' \cup \{\delta\}$ avec $(c.a \subseteq d) \in B'_+$, $B' \vdash c \leq c'$ et $d' <:_t d$. La base B'' telle que $B''_- = B'_-$ et $B''_+ = B'_+ \cup \{c'.a \subseteq d'\} - \{c.a \subseteq d\}$ est telle que $B'' \cup \{\delta\}$ est acceptable et B'' est donc une base contractée; de plus, $B'' \vdash B'_+$ (car $B'' \vdash c.a \subseteq d$, vu que $(c'.a \subseteq d') \in B''_+$, $B' \vdash c \leq c'$ et $d' <:_t d$) et $B'' \neq B'$ car $(c',d') \neq (c,d)$, donc B' n'est pas minimale, ce qui est contradictoire avec l'hypothèse.

(4') La démonstration de ce point est identique à celle du point précédent.

(5) Supposons que $(c'.a \subseteq c)$, $(o' \in c')$ et $(o'.a=o)$ soit acceptable dans $B' \cup \{\delta\}$. La base B'' telle que $B''_- = B'_-$ et $B''_+ = B'_+ \cup \{c'.a \subseteq c,\ o' \in c',\ o'.a=o\} - \{o \in c\}$ est telle que $B'' \cup \{\delta\}$

est acceptable. Soit B''‘=FP(B''), B''‘$\cup\{\delta\}$ est acceptable et B''‘ est sous forme première donc B''‘ est une base contractée; de plus, $B»' \vdash B'_+$ (car $B'' \vdash o \in c$, par la règle de ($\in$-inférence) et B''‘$\neq B'$, donc B' n’est pas minimale, ce qui est contradictoire avec l’hypothèse.

(6) Soit un ensemble d’assertions $\{c_i.a \subseteq d_{ij},\ o \in c_i\} \in \mathrm{Cn}(B)$ minimal tel que $\{v\}=\wedge d_{ij}$; par l’absurde, supposons que toutes ces assertions soient acceptables dans $B'\cup\{\delta\}$. Soit la base B'' telle que $B»=B'\cup\{c'.a \subseteq c,\ o' \in c',\ o'.a=o\}-\{o \in c\}$; $B''\cup\{\delta\}$ est acceptable. Soit B''‘=FP(B''), B''‘$\cup\{\delta\}$ est acceptable et B''‘ est sous forme première donc B''‘ est une base contractée; de plus, $B»' \vdash B'_+$ (car $B'' \vdash o.a=v$, par la règle de (o-valuation) et B''‘$\neq B'$, donc B' n’est pas minimale, ce qui est contradictoire avec l’hypothèse.

si) Soit B'' telle que $B'' \propto B'$ avec B' vérifiant les sept conditions, on va montrer que $B''=B'$. $B'' \propto B'$, donc $B''_- \subseteq B'_-$ par définition. Si $B'_+ \not\subset B''_+$, alors il existe $\mu \in B'_+$ et $\mu \notin B''_+$. $\mu \notin B''$ (car $\mu \notin B''_+$ et $\mu \notin (\mathrm{FP}(B)-B''_-)$ car $B'_+ \cap \mathrm{FP}(B)=\emptyset$ et $\mu \in B'_+$) et $B'' \vdash \mu$. Dans la suite de la démonstration, on montre que l’existence d’une telle assertion μ conduit à des contradictions, en découpant la démonstration selon le type d’assertion qu’est μ. Une fois cette démonstration faite, $B'_+ \subseteq B''_+$ sera prouvé et comme $B''_- \subseteq B'_-$, donc $B'=B''$ d’après la condition (1).

Soit μ tel que $\mu \in B'_+$, $\mu \notin B''$ et $B'' \vdash \mu$. Le principe de la démonstration repose sur le fait que vu que $\mu \in B'_+$, l’assertion n’appartient pas à la forme première et a donc été retirée de la clôture de B à cause de la présence d’autres assertions.

$c \leq c'$: $(c \leq c') \in B'_+$, donc pour toute chaîne $c1,\ldots,cn$ de classes, telle que $c=c1$, $c'=cn$ et $\forall i \in [1..n\text{-}1]\ (ci \leq ci+1) \in \mathrm{FP}(B)$ (car $(c \leq c') \in \mathrm{Cn}(B)$), il existe un ensemble I d’indices $\{i\}$ tel que $(ci \leq ci+1) \in B'_-$ (sinon $(c \leq c')$ est déductible de $\mathrm{FP}(B)-B'_-$ et ne peut donc pas appartenir à B'_+). De plus, $B'' \vdash c \leq c'$ et $(c \leq c') \notin B''$, donc il existe pour chaque $i \in I$, $(cj \leq ck) \in B''$ avec $1 \leq j \leq i$ et $i+1 \leq k \leq n$ (sachant qu’il peut s’agir de la même assertion pour plusieurs i).

S’il existe une assertion $(c_j \leq c_k)$ telle que $B' \nvdash c_j \leq c_k$, alors $(c_j \leq c_k)$ est inacceptable d’après le Lemme 14 car $B \vdash c_j \leq c_k$.

Si pour toutes les assertions $(c_j \leq c_k)$, $B' \vdash c_j \leq c_k$, alors $B' \vdash c_j \leq c'$ (une chaîne est reconstituée entre les deux classes), $(c \leq c') \in B'_+$:

si $c' \neq c_j$, $(c \leq c_j)$ est inacceptable dans $B' \cup \{\delta\}$ d'après la condition (2); or $B'' \vdash B'$ et $B'' \vdash c \leq c_j$ d'où $B'' \cup \{\delta\}$ est une base inacceptable et B'' ne peut donc être une base contractée de (B,δ).

si $c' = c_j$, $B' \vdash c \leq c_k$ et comme $(c \leq c') \in B'_+$ et $c' \neq c_k$ (si $c' = c_k$, $(c \leq c') \in B''$, ce qui est contraire aux hypothèses), $(c \leq c_k)$ est inacceptable dans $B' \cup \{\delta\}$ d'après la condition (2) et donc $B'' \vdash B'$ et $B'' \vdash c \leq c_j$ donc $B'' \cup \{\delta\}$ est une base inacceptable et ne peut être une base contractée de (B,δ).

$o \in c$: $(o \in c) \notin B''$ et $B'' \vdash o \in c$, donc, d'après les règles d'inférence, il existe au moins une classe $c' \neq c$ telle que $B'' \vdash c' \leq c$ et $(o \in c') \in B''_+$ ou il existe c', a et o' tels que $B'' \vdash c'.a \subseteq c$, $B'' \vdash o' \in c'$ et $B'' \vdash o'.a = o$.

dans le premier cas :

si $B' \vdash c' \leq c$, $(o \in c')$ n'est pas acceptable dans $B' \cup \{\delta\}$ (d'après la condition (3)); or $B'' \vdash B'$ et $B'' \vdash o \in c'$ donc $B'' \cup \{\delta\}$ est une base inacceptable et ne peut donc être une base contractée de (B,δ);

si $B' \nvdash c' \leq c$, $(c' \leq c)$ n'est pas acceptable dans $B' \cup \{\delta\}$ (d'après le Lemme 14); or $B'' \vdash B'$ et $B'' \vdash c' \leq c$ donc $B'' \cup \{\delta\}$ est une base inacceptable et ne peut donc être une base contractée de (B,δ).

dans le second cas : on obtient immédiatement en utilisant le point (5) que l'une de ces trois assertions n'est pas une assertion acceptable dans $B' \cup \{\delta\}$ et donc que B'' n'est pas une base contractée de (B,δ).

$c.a \subseteq d$: $(c.a \subseteq d) \in B'_+$ donc il existe un ensemble d'ensembles d'assertions $\{(c_i.a \subseteq d_i) \in \mathrm{FP}(B),\ B \vdash c \leq c_i\}$ avec $d = \wedge d_i$ (car $(c.a \subseteq d) \notin \mathrm{FP}(B)$); $B'' \vdash (c.a \subseteq d)$, mais $(c.a \subseteq d) \notin B''$, donc il existe au moins un ensemble $\{B'' \vdash c_i.a \subseteq d_i$ et $B'' \vdash c \leq c_i\}$ et pour cet ensemble, il existe i tel que $B' \nvdash c_i.a \subseteq d_i$ ou $B' \nvdash c \leq c_i$ (sinon $(c.a \subseteq d)$ ne peut appartenir à la forme première de la base)

s'il existe i tel que $B' \nvdash c \leq c_i$ alors $(c \leq c_i)$ est inacceptable dans $B' \cup \{\delta\}$ (d'après le Lemme 14) et donc, comme $B'' \vdash c \leq c_i$ et que $B'' \vdash B'$, $B'' \cup \{\delta\}$ est une base inacceptable et ne peut donc être une base contractée de (B,δ);

s'il existe i tel que $B' \nvdash c_i.a \subseteq d_i$ alors $(c_i.a \subseteq d_i)$ est inacceptable dans $B' \cup \{\delta\}$ (d'après le Lemme 15 ou le Lemme 16), $B'' \cup \{\delta\}$ est une base inacceptable et ne peut donc être une base contractée de (B,δ).

$o.a=v$: $(o.a=v) \in B'_+$, donc il existe un ensemble d'ensembles d'assertions $\{B \vdash c_i.a \subseteq d_{ij}, B \vdash o \in c_i\}$ avec $\{v\} = \wedge d_{ij}$. Il existe au moins l'un de ces ensembles tel que $\{B'' \vdash c_i.a \subseteq d_{ij}, B'' \vdash o \in c_i\}$ (car $(o.a=v) \notin B''_+$ et $B'' \vdash o.a=v$ et $Cn(B'') \subseteq Cn(B)$). Or d'après le point (6), l'une de ces assertions au moins n'est pas acceptable dans $B' \cup \{\delta\}$, donc $B'' \cup \{\delta\}$ est inacceptable et n'est pas une base contractée, donc il n'existe pas d'assertion du type $(o.a=v)$ telle que $(o.a=v) \in B'_+$ et $(o.a=v) \notin B''_+$. ◊

PROPOSITION 24 : SATISFACTION DES POSTULATS (AGM)
Les postulats (AGM) (G2-6) sont satisfaits.

démonstration.

Ces postulats sont démontrés pour toute base révisée B' de (B,δ); B'' est la base contractée associée.

(G2) : Immédiat, car B' est définie par union de B'' et de δ.

(G3) : Par définition de la base contractée, $Cn(B'') \subset Cn(B)$, donc $Cn(B' \cup \{\delta\}) \subset Cn(B \cup \{\delta\})$

(G4) : La révision n'est définie que dans le cas où l'assertion est inacceptable dans la base, dans le cas où elle est acceptable elle est ajoutée à la base.

(G5) : Si l'assertion est cohérente, la base révisée est acceptable.

(G6) : Deux assertions du langage d'objets présenté ne peuvent être équivalentes que si elles sont égales et le résultat est alors immédiat.

(G1) n'est pas satisfait car la base révisée obtenue n'est pas fermée par déduction, mais sous forme première.

(G7) et (G8) n'ont pas de sens dans notre contexte, car il n'y a pas d'opérateur $\wedge$ et que les ajouts d'assertion sont traités les uns après les autres. ◊

J. Forme première, déduction, acceptabilité et complexité pour le langage simple

Cette annexe donne d'une part les algorithmes permettant de trouver la forme première d'une base, de vérifier si une assertion est une conséquence d'une base et de déterminer si une assertion est acceptable dans une base. Ces algorithmes sont surtout donnés dans le souci de déterminer leur complexité. D'autre part, la complexité des algorithmes de révision est donnée.

Cette section étudie le cas du langage simple, la section suivante le cas du langage avec références aux instances.

Soit une base B, définie par un ensemble d'assertions des quatre types. Soient n_c, n_o, n_d et n_v respectivement le nombre d'assertions contenues dans la base de la forme $(c \leq c')$, $(o \in c)$, $(c.a \subseteq d)$ et $(o.a=v)$. Nous allons déterminer la complexité de la détermination de la forme première, de la déduction et des algorithmes de révision.

1. Détermination de la forme première

a) Schéma de l'algorithme

L'idée est de ne pas utiliser la clôture déductive de la base. Pour cela, à partir de la base initiale, on détermine les assertions de la forme $(c \leq c')$, $(o \in c)$ et $(o.a=v)$ qui doivent être conservées dans la forme première et on calcule les restrictions d'attribut de domaine pour chacune des classes, puis on supprime celles qui sont redondantes.

La forme normale de la base est une étape intermédiaire intéressante : base dans laquelle chaque classe possède la restriction d'attribut minimale et les attributs d'instance qui admettent une valeur dans la clôture déductive de la base sont valués. Le calcul de cette forme normale peut se décomposer en plusieurs étapes, qui seront considérées comme autant de procédures :

- obtention des assertions $(c \leq c')$: les assertions qui appartiennent à la forme première sont dans la base initiale (en effet dans la clôture déductive, on ne peut avoir que des classes attachées à des classes moins spécifiques), il suffit donc de supprimer celles qui sont superflues (Cred)
- obtention des assertions $(o \in c)$: pour la même raison que ci-dessus, il suffit de supprimer les rattachements redondants (Ored)
- obtention du domaine minimal pour chaque couple (c, a) : il est obtenu par intersection de toutes les restrictions posées sur la classe et ses super-classes (DNorm)

La forme normale s'exprime alors par : FN(B)=DNorm(ORed(CRed B))

Rappelons que l'intérêt de la forme première par rapport à cette forme normale est sa minimalité, c'est-à-dire la propriété selon laquelle une assertion supprimée ne peut plus être déduite (ce qui n'est pas le cas dans la forme normale car si une classe et sa super-classe ont la même restriction d'attribut, la suppression de la restriction dans la classe pourra être redéduite).

Pour la forme première :

- obtention des assertions ($o.a=v$) : il s'agit de supprimer dans la base, les assertions de ce type qui peuvent être déduites, vu les restrictions de domaine posées sur les classes d'appartenance des classes (d'où l'importance d'avoir la restriction de domaine dans chacune des classes) (VRed)
- obtention des assertions ($c.a\subseteq d$) : il s'agit de supprimer celles qui sont redondantes, c'est-à-dire qui peuvent être redéduite par intersection des restrictions sur les super-classes (DRed).

Elle s'exprime alors par : FP(B)=DRed(VRed(FN(B)))

Pour chaque opération, la complexité en temps (selon le nombre de comparaisons effectuées) et la taille de la base résultant seront données *dans le pire cas* (en particulier la taille sera déclarée inchangée même si elle peut être restreinte).

On suppose que la complexité des opérations relatives aux types (vérification de l'inclusion de deux domaines, de l'égalité de deux valeurs, de l'appartenance d'une valeur à un domaine, l'intersection de domaines) sont constantes.

b) Structures de données

Afin de diminuer le coût d'accès à certaines information, nous définissons des structures (hash-tables) qui permettent de stocker ces informations lors de leur premier calcul et de pouvoir ensuite y accéder en temps constant :

On crée des hash-tables donnant :

T1 : pour un couple (instance, attribut), la valeur de l'attribut

T2 : pour un couple (classe, attribut), la restriction de domaine

T3 : pour une classe, l'ensemble des super-classes directes, des super-classes et de sous-classes

T4 : pour une instance, l'ensemble des classes d'appartenance, ainsi que l'ensemble des plus basses

On indiquera entre parenthèses les tables utilisées dans les descriptions ci-dessous.

c) Procédure CRed

Obtenir les assertions du type ($c \leq c'$) : seules les assertions telles que il n'existe pas c'' telle que : ($c \leq c''$) appartient à la base et il existe un chemin entre c'' et c' sont conservées.

temps :	pour chaque c telle ($c \leq c') \in B$	n_c
	déterminer si c' est une classe minimale,	
	c'est-à-dire s'il existe des classes entre c et c'.	n_c^2
	(et T3 est construite)	
soit	$\mathbf{o(n_c^3)}$	
taille :	inchangée	

d) Procédure ORed

Obtenir les assertions du type ($o \in c$) : seules les assertions telles qu'il n'existe pas c' telle que : ($o \in c'$) appartient à la base et il existe un chemin entre c' et c sont conservées.

temps :	pour chaque o	$\lvert O \rvert$
	pour l'ensemble de classes contenant o ($(o \in c) \in B$)	n_o
	déterminer les classes les plus basses	n_c^2
	(et T4 est construite)	
soit	$\mathbf{o(\lvert O \rvert * n_o * n_c^2)}$	
taille :	inchangée	

e) Procédure DNorm

Obtenir la restriction de domaine minimale dans chaque classe pour chaque attribut

temps :	pour chaque classe	$\lvert C \rvert$
	pour toutes les super-classes (utilisation de T3)	$\lvert C \rvert$
	pour chaque attribut	$\lvert A \rvert$
	obtenir les restrictions de domaine	n_d
	puis on fait l'intersection	cste
	(et T2 est construite)	
soit	$\mathbf{o(\lvert C \rvert^2 * \lvert A \rvert * n_d)}$	
taille :	$n'_d = \lvert C \rvert * \lvert A \rvert$	

f) Procédure DRed

Obtenir les restrictions d'attribut qui appartiennent à la forme première, en supprimant toutes celles qui peuvent être déduites par intersection des domaines des super-classes

temps :	pour chaque couple (classe, attribut)	$\lvert C \rvert * \lvert A \rvert$
	pour toutes les super-classes directes (utilisation de T3)	$\lvert C \rvert$
	chercher leur restriction d'attribut (utilisation de T2)	cste
	puis faire l'intersection	cste

si cette intersection est égale à la restriction de domaine de la classe, celle-ci est supprimée de la base, dans le cas contraire elle est conservée.

soit $\mathbf{o(|C|^2*|A|)}$

taille : inchangée

g) Procédure VRed

Obtenir les valeurs des attributs des instances; il s'agit de supprimer toutes celles qui peuvent être déduites

temps :	pour chaque assertion de type ($o.a=v$)	n_v
	pour toutes les classes d'appartenance les plus spécifiques (utili° de T4)	$\|C\|$
	obtenir leur restriction de domaine (pour l'attribut considéré) (utilis° de T2)	cste
	faire l'intersection de tous ces domaines	cste

si l'intersection est égale à la valeur, l'assertion est supprimée, conservée sinon.

soit $\mathbf{o(|C|*n_v)}$

taille : inchangée

h) Conclusions

Procédures	Complexité
CRed	$o(n_c^3)$
ORed	$o(\|O\|*n_o*n_c^2)$
DNorm	$o(\|C\|^2*\|A\|*n_d)$
DRed	$o(\|C\|^2*\|A\|)$
VRed	$o(\|C\|*n_v)$
Forme normale	$o(\max(n_c^3, \|O\|*n_o*n_c^2, \|C\|^2*\|A\|*n_d)$
Forme première	$o(\max(n_c^3, \|O\|*n_o*n_c^2, \|C\|^2*\|A\|*n_d, \|C\|*n_v)$

Tableau 3 : complexité du calcul de le forme normale et de la forme première

PROPOSITION 14 : COMPLEXITÉ DU CALCUL DE LA FORME PREMIÈRE

L'algorithme de calcul de la forme première a une complexité polynomiale en nombre d'assertions de la base.

2. Déduction

La première étape consiste à calculer la forme première de la base.

Puis suivant le type d'assertion à déduire :

a) **$(o \in c)$**

pour chaque classe c' d'appartenance de o (utilisation de T4) $|C|$
chercher s'il existe un chemin entre c' et c (utilisation de T3) $|C|$
soit **$o(|C|^2)$**

b) **$(c \leq c')$**

pour chaque super-classe c'' de c (utilisation de T3) $|C|$
chercher s'il existe un chemin entre c'' et c' (utilisation de T3) $|C|$
soit **$o(|C|^2)$**

c) **$(o.a=v)$**

vérifier s'il existe une valeur pour $o.a$ cste
(si oui et si cette valeur est v, vrai, si cette valeur n'est pas v, faux)
pour chaque classe d'appartenance de o(utilisation de T4) $|C|$
chercher la restriction d'attribut (utilisation T2) cste
faire l'intersection des domaines cste
(si cette intersection est égale à la valeur v alors vrai, sinon faux)
soit **$o(|C|)$**

d) **$(c.a \subseteq d)$**

chercher le domaine correspondant à $c.a$ cste
vérifier que d contient le domaine trouvé cste
soit **o(1)**

Opérations de déduction	Complexité
$(o \in c)$	$o(\|C\|^2)$
$(c \leq c')$	$o(\|C\|^2)$
$(o.a=v)$	$o(\|C\|)$
$(c.a \subseteq d)$	o(1)
déduction uniquement	$o(\|C\|^2 * \|A\|)$
déduction avec calcul de la forme première	$o(\max(n_c^3, \|O\| * n_o * n_c^2, \|C\|^2 * \|A\| * n_d, \|C\| * n_v)$

Tableau 4 : complexité de la déduction

PROPOSITION 5 : COMPLEXITÉ DE LA DÉDUCTION

L'algorithme de déduction a une complexité polynomiale en nombre d'assertions de la base.

3. Test d'acceptabilité (consistance, cohérence)

La vérification de la cohérence ne nous est utile que lors d'un ajout d'une assertion, donc on ne vérifie pas l'acceptabilité d'une base quelconque, mais l'acceptabilité d'une base,

précédemment acceptable, suite à l'ajout d'une assertion.[21] Cette vérification se fait à l'aide de la forme normale, afin d'avoir directement la restriction de domaine posée sur les classes d'appartenance des instances et les valeurs des attributs des instances si de telles valeurs existent.

Suivant le type d'assertion ajoutée à la base, étudions la procédure de détection et la complexité.

a) $(o \in c)$

si l'assertion ne peut être déduite	$\|C\|^2$	
pour chaque attribut	$\|A\|$	
trouver la valeur éventuelle de *o.a* direct (T1)	cste	
pour toutes ses classes d'appartenance (T4) et *c*	$\|C\|$	
trouver leur domaine (T2)		cste
et faire l'intersection		
si l'intersection est vide cas 3 d'inconsistance		
si la valeur directe n'appartient pas à l'intersection, cas 2 d'inconsistance		
mise à jour de T4	$\|C\|^2$	

soit $\mathbf{o(|C|*(|A|+|C|))}$

b) $(c \leq c')$

pour l'incohérence :		
pour chaque attribut	$\|A\|$	
pour chaque sous-classe (T3)	$\|C\|$	
re-calcul des domaines des sous-classes (T2)	cste	
+ mise à jour de T2		
vérification que les domaines sont non vides	cste	
pour l'inconsistance :		
pour chaque attribut	$\|A\|$	
pour chaque instance	$\|O\|$	
vérifier si elle est attachée à une sous-classe de *c*	$\|C\|$	
puis récupérer les restrictions d'attribut (nouvelle T2)	$\|C\|$	
et la valeur directe éventuelle de *o.a*	cste	
puis faire les vérifications données dans le Tableau 5	cste	
Mise à jour de T3 : dans chaque super-classe de *c'*, mettre les sous-classes de *c*	$\|C\|*\|C\|$	
et dans chaque sous-classe de *c*, mettre les super-classes de *c'*	$\|C\|*\|C\|$	

[21] Dans le cas de la vérification de l'acceptabilité d'une base, la complexité s'évalue en supposant que la base est reconstruite en ajoutant chacune de ses assertions, soit une multiplication par la taille de la base de la complexité.

	intersection vide	une valeur	plusieurs valeurs
valeur dans la FP	cas 3	si différent : cas 1 si égal : OK	si contenue : OK sinon : cas 2
pas de valeur dans la FP	cas 3	OK	OK

Tableau 5 : tableau indiquant s'il y a consistance ou le cas d'inconsistance dans lequel on se trouve suivant l'existence ou non d'une valeur pour *o.a* et l'intersection de domaine des classes d'appartenance.

Ici, le cas 1 est un sous cas du cas 2, mais ce n'est pas toujours le cas (en particulier lorsque l'utilisateur a donné explicitement deux valeurs pour un attribut d'instance).

soit $\mathbf{o(|C|*(|O|*|A|+|C|))}$

c) $(o.a=v)$

vérifier que *B* ne permet pas de déduire une valeur pour *o.a*	cste+ $\|C\|^2*\|A\|$
puis, pour toutes les classes d'appartenance de *o* (T4)	$\|C\|$
comparer avec la restriction posée (T2)	cste
mise à jour de T1	cste

soit $\mathbf{o(|C|^2*|A|)}$

d) $(c.a \subseteq d)$

pour l'incohérence :

dans toutes les sous-classes (T3)	$\|C\|$
comparer d avec les domaines précédents (T2)	cste
mise à jour de T2	cste

pour l'inconsistance :

pour toutes les instances	$\|O\|$
pour leur ensemble de classe d'attachement (T4)	$\|C\|$
récupérer les restrictions de domaine (T2)	cste
récupérer la valeur éventuelle de l'attribut de l'instance (T1)	cste
l'inconsistance est détectée selon le Tableau 5	cste

Suivant l'existence ou non d'une valeur d'attribut de l'instance et l'intersection de tous les domaines des classes d'appartenance

soit $\mathbf{o(|O|*|C|)}$

Acceptabilité	Complexité
$(o \in c)$	$o(\|C\|*(\|A\|+\|C\|)))$
$(c \leq c')$	$o(\|C\|*(\|O\|*\|A\|+\|C\|))$
$(o.a=v)$	$o(\|C\|^2*\|A\|)$
$(c.a \subseteq d)$	$o(\|O\|*\|C\|)$
acceptabilité	$o(max(\|O\|*\|A\|*\|C\|, \|C\|^2*\|A\|)$

Tableau 6 : complexité du test d'acceptabilité

4. Révision

Pour chaque partie de l'algorithme (correspondant aux ajouts d'assertion des quatre types), le coût de chacune des solutions élémentaires et de l'obtention de l'ensemble des bases révisées est donné. Il faut noter ici que le fait que l'on ne fournisse pas à l'utilisateur toutes les bases révisées, mais que des points de choix interviennent diminue nettement la complexité. Nous allons donner cette complexité également.

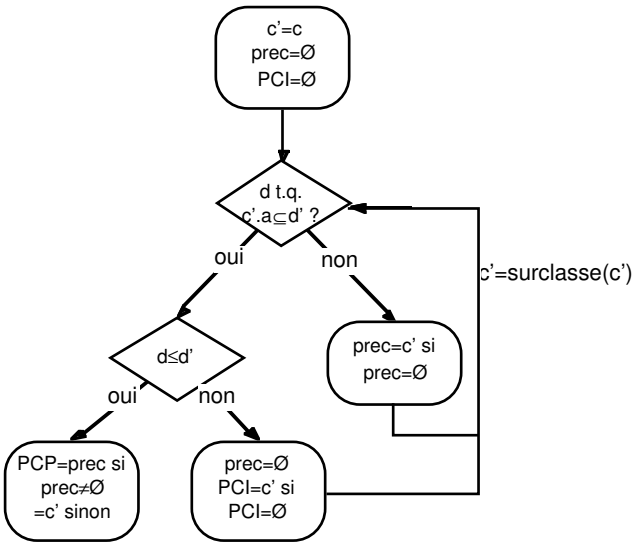
Cela permettra de montrer que l'algorithme a un coût polynomial. Les coûts des modifications élémentaires sont données en premier lieu.

Modification élémentaire	Complexité
Suppression d'une valeur d'attribut d'une instance	cste
Déplacer une instance sous une super-classe	$\|C\|$
Déplacer une classe sous une super-classe	$\|C\|*\|A\|*\|O\|$
Remonter des instances	$\|O\|$
Agrandir une restriction d'attribut	$\|C\|$
PCP et PCI	$\|C\|$

Tableau 7 : complexité des modifications élémentaires

Quelques précisions :

(1)PCP(c,a,d) et PCI(c,a,d) sont calculés en un coût $|C|$, selon l'algorithme suivant :



(2)Pour pouvoir déterminer les assertions du type (*o.a=v*) qui sont perdues au cours des modifications de la base et qui sont donc à ajouter pour assurer la minimalité des bases révisées, il s'agit de trouver les nouvelles restrictions de domaines agissant sur les instances :

pour un attribut *a*, et une instance *o*, tel qu'il existe *v* tel que (*o.a=v*) n_v
trouver l'ensemble des restrictions qui pesaient avant sur cette instance (T4 et T2) $|C|$
puis trouver les restrictions perdues :
si la modification de la base est :

changer *o* de classe (récupérer les restrictions entre *c* et *c'*, T2)[22] $|C|$
modifier des restrictions d'attribut (récupérer les anciennes restrictions) $|C|$
changer la super-classe d'une classe
(récupérer les restrictions de domaine entre l'ancienne super-classe et la nouvelle $|C|$

Si l'ancienne intersection est réduite à une valeur et pas la nouvelle, l'assertion doit être ajoutée dans la base révisée.

[22] A n'est pas compté ici car il est déjà pris en compte au niveau de nv.

$\delta=(o.a=v)$		nb possibilités
solution 1	cste	
solution 2	$\|C\|$	
solution 3	$\|C\|*\|A\|*\|O\|$	$\|C\|$
solution 4	$\|C\|*\|A\|*\|O\|*\|O\|$	$\|C\|$
solution 5	$\|C\|$	$\|C\|$
obtenir toutes les bases	$\|C\|^2*\|A\|*\|O\|^2$	
obtenir une base	$\|C\|*\|A\|*\|O\|^2$	

Il y a un seul choix de l'utilisateur entre 2, 3, 4 et 5.

$\delta=(o\in c)$		nb possibilités (une par défaut)
PCI et PCP pour tous les attributs	$\|C\|*\|A\|$	
solution 1'	$\|C\|*\|A\|*\|O\|$	$\|C\|*\|A\|*\|C\|*\|A\|$
solution 2	$\|C\|$	
solution 3	cste	
obtenir toutes les bases	$\|C\|^3*\|A\|^3*(\|A\|+\|O\|)$	
obtenir une base	$\|C\|*\|A\|*\|O\|$	

La complexité de l'obtention de toutes les bases est donnée dans le tableau ci-dessous.

	obtenir une base	obtenir toutes les bases
1. Calcul de PCP et PCI	$\|C\|*\|A\|$	$\|C\|*\|A\|$
2. choisir une classe dans PCP et l'une dans PCI	1	$*\|C\|*\|A\|*\|C\|*\|A\|$
3. partitionner les attributs toujours problématiques en I2 et I3	1	$2^{\|A\|}*$
4. solution 1' sur I1 et I2	$\|C\|*\|A\|*\|O\|$	$\|C\|*\|A\|*\|O\|$
5. solution 2 sur I2	$\|C\|*\|A\|$	$\|C\|*\|A\|$
6. solution 3 sur I3	$\|A\|$	$\|A\|$
total	$\|C\|*\|A\| + \|A\| + \|C\|*\|A\|*\|O\| + \|C\|*\|A\| + \|A\| = \|\mathbf{C}\|*\|\mathbf{A}\|*\|\mathbf{O}\|$	$\|C\|*\|A\| + \|C\|*\|A\|*\|C\|*\|A\| *2^{\|A\|}*(\|C\|*\|A\|*\|O\| + \|C\|*\|A\| + \|A\|) = \mathbf{2^{\|A\|}*\|C\|^3*\|A\|^3*\|O\|}$

$\delta=(c.a\subseteq d)$		nb possibilités (une par défaut)
PCI et PCP	$\|C\|$	
solution 1	$\|C\|$	
solution 2	$\|C\|*\|A\|*\|O\|$	
solution 2'	$\|C\|*\|A\|*\|O\|$	
solution 3	$\|C\|^2*\|A\|*\|O\|$	
solution 4	$\|C\|*\|A\|*\|O\|$	
solution 5	$\|C\|^2*\|A\|*\|O\|$	
solution 6	$\|A\|*\|O\|$	
solution 7	cste	
solution 8	$\|C\|$	
obtenir toutes les bases	$\|C\|^2*\|A\|*\|O\|*3^{\|C\|}+\|C\|*2^{\|O\|}$	
obtenir une base	$\|C\|^3*\|A\|*\|O\|$	

S1, S2, S2' ou S3 pour *c*	cste
pour les classes n'ayant pas de super-classes dans *C* : S4, S5 ou S6	$3^{\|C\|}$
pour les instances : S7 ou S8	$2^{\|O\|}$
Total	$3^{\|C\|}*2^{\|O\|}$

Le coût de l'algorithme de révision pour obtenir une base révisée est polynomial. Malgré l'interactivité avec le concepteur, la complexité aurait pu être polynomiale, en particulier si l'un des choix faits à l'utilisateur avait nécessité un calcul de complexité polynomiale. Vu que le nombre de bases révisées minimales est dans certains cas exponentiel, l'obtention de toutes les bases révisées est exponentielle.

K. Forme première, déduction, acceptabilité et complexité pour le langage avec référence à des instances

1. Détermination de la forme première

a) Schéma de l'algorithme

Par rapport au cas simple, il y a trois règles de plus dans le système déductif (c-héritage, c-clôture et -$\in$ inférence) et les règles symétriques dans le calcul de la forme première. Soient n'_c,

et n'$_o$ respectivement le nombre d'assertions contenues dans la base de la forme ($c.a \subseteq c'$) et ($o.a=o'$). Comme précédemment, nous allons calculer la forme normale et la forme première par une succession de procédures.

Pour le calcul de la forme normale, les étapes sont les suivantes :

- obtention des assertions ($c \leq c'$) CRed
- obtention des assertions ($o \in c$) : Ored - on supprime les rattachements aux classes moins spécifiques
- obtention des assertions ($c.a \subseteq d$) pour chaque couple (c,a) : DNorm
- obtention des assertions ($c.a \subseteq c'$) : de la même manière que DNorm mais pour les restrictions par des classes (CHerit)
- calcul des ($o \in c$) de la clôture déductive : par la règle de ($\in$-inférence), on obtient toutes les assertions de rattachement d'instances à des classes (Osat)

Pour obtenir la forme première, à partir de la forme normale, les étapes sont les suivantes :

- obtention des assertions ($o.a=v$) : Vred en utilisant les rattachements d'instances calculés dans la forme première.
- obtention des assertions ($o \in c$) : on supprime les assertions de la base initiale qui peuvent se déduire par la règle de ($\in$-inférence) (ce sont donc celles qui apparaissaient dans la base initiale mais qui peuvent être déduites au vu des restrictions d'attribut des classes. (O-Anti-Herit)
- obtention des assertions ($c.a \subseteq d$) : Dred
- obtention des assertions ($c.a \subseteq c'$) : suppression de celles qui sont redondantes (car déductibles par héritage des contraintes des super-classes) (CAnti-Herit)

Ci-dessous sont détaillées uniquement les procédures qui diffèrent de celles données ci-dessus pour le langage simple.

b) Structures de données

On reprend les mêmes hash-tables, mais dans T1, on indique également la valeur d'attribut d'une instance s'il s'agit d'une instance; de même dans T2, on indique également les restrictions par des classes.

c) CHerit

Cette procédure est identique à Dnorm, la complexité étant alors : $\mathbf{o(|C|^2*|A|*n'_c)}$

d) Osat

En fait, vu que ces assertions ne sont utiles que pour les procédures suivantes mais ne devront pas apparaître dans la forme normale, on les met dans une hash-table : a chaque instance est associé l'ensemble de ses classes d'appartenance.

Ordonner les instances par ordre de référence (pour obtenir une liste d'instances tels que chaque instance ne référence pas les instances placées avant lui dans la liste) $|\mathbf{O}|*|\mathbf{A}|$

schéma de l'algorithme :

pour toutes les assertions $o.a=o'$, incrémenter la marque de o'.

trouver les instances ayant marque à 0 en décrémentant ceux qu'elles référençaient.

(au plus $|O|$ éléments dans la liste)

en parcourant la liste d'instances créées,
obtenir toutes les instances qu'elle référence pour un ensemble d'attribut $|\mathbf{O}|$
les classes qui restreignent ses classes d'appartenance pour chaque attribut $|\mathbf{C}|$
attacher les instances référencées dans les classes correspondantes
(mise à jour de T4)
soit $\mathbf{o(|O|*(|A|+|C|))}$

e) O-Anti-Herit

Pour chaque assertion $(o \in c)$ de la base $|\mathbf{O}|$
trouver les assertions $(o'.a=o)$ $\mathbf{n'_o}$
trouver les classes c' qui restreignent l'attribut a des classes d'appartenance de o' $|\mathbf{C}|$
Si $c' \leq c$, alors ôter $(o \in c)$, sinon la conserver $|\mathbf{C}|$

(mise à jour de T4 pour conserver les plus basses classes)

soit $\mathbf{o(|O|*n'_c*|C|^2)}$

f) c-Anti-Herit

Procédure similaire à Dred, sauf que seul l'héritage des super-classes est vérifié et non l'intersection.

soit $\mathbf{o(|C|^2*|A|)}$

g) Conclusions

Procédure	Complexité
CRed	$o(n_c^3)$
ORed	$o(\|O\|*n_o*n_c^2)$
DNorm	$o(\|C\|^2*\|A\|*n_d)$
CHerit	$o(\|C\|^2*\|A\|*n'_c)$
OSat	$o(\|O\|*(\|A\|+\|C\|))$
VRed	$o(\|C\|*n_v)$
O-Anti-Herit	$o(\|O\|*n'_c*\|C\|^2)$
DRed	$o(\|C\|^2*\|A\|)$
C-Anti-Herit	$o(\|C\|^2*\|A\|)$
Forme normale	$o(\max(n_c^3, \|O\|*n_o*n_c^2, \|C\|^2*\|A\|*n'_c, \|C\|^2*\|A\|*n_d)$
Forme première	$o(\max(n_c^3, \|O\|*n_o*n_c^2, \|C\|^2*\|A\|*n_d, \|C\|*n_v)$

Tableau 8 : complexité du calcul de la forme normale et de la forme première pour le langage avec référence aux instances

La complexité du calcul de la forme première est donc également polynomiale dans le cas du langage avec références aux instances.

2. Déduction

La déduction de $(o \in c)$ est identique au cas précédent si on utilise les appartenances aux classes qui sont déductibles et qui appartiennent donc à la forme normale. La déduction de $(c \leq c')$, $(o.a=v)$ et $(c.a \subseteq d)$ est identique au cas précédent.

La déduction de $(o.a=o')$ s'effectue en temps constant (car o' est dans la table).

La déduction de $(c.a \subseteq c')$:

récupérer toutes les classes telles que $(c.a \subseteq c'')$ dans la forme normale	$\|C\|$
vérifier que l'une est telle que $c'' \leq c'$	$\|C\|$

Opérations de déduction	Complexité
$(o \in c)$	$o(\|C\|^2)$
$(c \leq c')$	$o(\|C\|^2)$
$(o.a=v)$	$o(\|C\|^2 * \|A\|)$
$(o.a=o')$	$o(1)$
$(c.a \subseteq d)$	$o(1)$
$(c.a \subseteq c')$	$o(\|C\|^2)$
déduction	$o(\|C\|^2 * \|A\|)$
déduction avec calcul de la forme première	$o(\max(n_c^3, \|O\| * n_o * n_c^2, \|C\|^2 * \|A\| * n_d, \|C\| * n_v)$

Tableau 9 : complexité de la déduction

La complexité de la déduction est donc identique pour le langage simple et pour le langage complexe.

3. Test d'acceptabilité (consistance, cohérence)

Il faut à présent tenir compte de la règle de ($\in$-inférence) en particulier, ce qui affecte toutes les assertions (dans le sens où à cause de cette règle, des inconsistances peuvent se produire).

Evaluons tout d'abord la complexité de deux opérations qui seront utiles pour la suite.

(1) Est-ce qu'une classe est cohérente avec un ensemble de classes $\{c$?

(2) Est-ce qu'une instance o peut appartenir à un ensemble de classes $\{c\}$?

(1)

Pour chaque attribut	$\|A\|$
récupérer la restriction de c et des autres classes	$\|C\|$
si domaines, vérifier que l'intersection est non vide	cste
si classes, vérifier pour tous les attributs	$\|C\|*\|A\|$

En effet, il peut y avoir une chaîne de classes qui se référencent par un chemin d'attribut, mais vu qu'il n'y a pas de cycles, il y a au maximum $|C|$ classes qui seront touchées, avec vérification à ce moment-là d'au maximum $|A|$ attributs qui sont restreints par des domaines.

Soit **c1=** $\mathbf{|C|*|A|}$.

(2)

Pour chaque attribut	$\|A\|$
récupérer la valeur éventuelle de $o.a$	cste
récupérer les restrictions d'attributs des classes d'appartenance	$\|C\|$
si restriction par des domaines	cste
si restriction par des classes	$\|C\|*\|A\|$

soit **c2=** $\mathbf{|C|*|A|}$

a) $(o \in c)$

si l'assertion ne peut être déduite (T4, T3)	$\|C\|^2$
pour chaque attribut	$\|A\|$
trouver la valeur éventuelle de *o.a* direct (T1)	cste
pour toutes ses classes d'appartenance (T4) et *c*	$\|C\|$
trouver leur restriction (T2)	cste
si ce sont des domaines, faire l'intersection	
si l'intersection est vide cas 3 d'inconsistance	
si la valeur directe n'appartient pas à l'intersection, cas 2 d'inconsistance	
si ce sont des classes et si *o.a* a une valeur, proc. (2) avec cette instance	c2
mise à jour de T4	$\|C\|^2$

soit $\mathbf{o(|C|*(|A|+|C|))}$

b) $(c \leq c')$

pour l'incohérence :

pour chaque attribut	$\|A\|$
pour chaque sous-classe (T3)	$\|C\|$
si restreints par des domaines recalcul des domaines des sous-classes (T2) + mise à jour de T2	cste
vérification que les domaines sont non vides	cste
si restreints par des classes, proc. (1) appliquée sur la restriction de *c.a*	c1

pour l'inconsistance :

pour chaque attribut	$\|A\|$
pour chaque instance	$\|O\|$
vérifier si elle est attachée à une sous-classe de *c*	$\|C\|$
puis récupérer les restrictions d'attribut (nouvelle T2)	$\|C\|$
et la valeur directe éventuelle de *o.a*	cste
puis, si domaine, faire les vérifications données dans le Tableau 5	cste
si restrictions par des classes, appliquer la proc. (2) sur la valeur de *o.a*	c
Mise à jour de T3 : dans chaque super-classe de *c'*, mettre les sous-classes de *c*	$\|C\|*\|C\|$
et dans chaque sous-classe de *c*, mettre les super-classes de *c'*	$\|C\|*\|C\|$

Ici, le cas 1 est un sous cas du cas 2, mais ce n'est pas toujours le cas (en particulier lorsque l'utilisateur a donné explicitement deux valeurs pour un attribut d'instance).

soit $\mathbf{o(|C|*(|O|*|A|+|C|))}$

c) $(o.a = v)$

si *v* est une valeur de type,	
vérifier que *B* ne permet pas de déduire une valeur pour *o.a*	cste+ $\|C\|^2*\|A\|$
puis, pour toutes les classes d'appartenance de *o* (T4)	$\|C\|$
comparer avec la restriction posée (T2)	cste
si *v* est une instance	
vérifier que *B* ne permet pas de déduire une valeur pour *o.a* (pas possible par inférence)	cste
appliquer proc. (2) pour *v* et les classes d'appartenance de *o*	c2+$\|C\|$
mise à jour de T1	cste

soit **o(**$|\mathbf{C}|^{2}*|\mathbf{A}|$**)**

d) **(*c.a*⊆*d*)**

pour l'incohérence :

dans toutes les sous-classes (T3)	$\|C\|$
si d est un domaine, comparer d avec les domaines précédents (T2)	cste
si *d* est une classe, appliquer proc. (1) sur *d* et les autres classes restriction	c1
mise à jour de T2	cste

pour l'inconsistance :

pour toutes les instances qui sont attachées à une sous-classe	$\|O\|*\|C\|$
si restriction par domaine, pour leur ensemble de classe d'attachement (T4)	$\|C\|$
récupérer les restrictions de domaine (T2)	cste
récupérer la valeur éventuelle de l'attribut de l'instance (T1)	cste
l'inconsistance est détectée selon le Tableau 5	cste
si restriction par des classes, appliquer proc. (2) sur chaque instance	c2

Suivant l'existence ou non d'une valeur d'attribut de l'instance et l'intersection de tous les domaines des classes d'appartenance

soit **o(**$|\mathbf{C}|*(|\mathbf{O}|+|\mathbf{A}|)$**)**

Acceptabilité	Complexité
(*o*∈ *c*)	o($\|C\|*(|A|+|C|)$))
(*c*≤*c'*)	o($\|C\|*(|O|*|A|+|C|)$)
(*o.a*=*v*)	o($\|C\|^{2}*|A|$)
(*c.a*⊆*d*)	o($\|C\|*(|O|+|A|)$)
acceptabilité	o(max($\|O\|*|A|*|C|$, $\|C\|^{2}*|A|$)

Tableau 10 : complexité du test d'acceptabilité

4. Révision

Nous allons donner la complexité de chaque modification élémentaire, puis celle des solutions correspondant à chaque type d'assertion ajoutée. Les complexités pour obtenir une solution et toutes les solutions seront données.

a) Modifications élémentaires

(1) Suppression d'une valeur d'un attribut d'instance

L'hypothèse est que (*o.a*=*v*)∈ FP(*B*) ou (*o.a*=*o'*)∈ FP(*B*)

S(*o*,*a*,*v*)	cste

S(*o*,*a*,*o'*), pour obtenir B'_{+}, il faut trouver toutes les assertions du type (*o'*∈ *c'*) qui doivent y être ajoutées :

Récupérer toutes les classes *c'* telles que (*o*∈ *c*) et (*c.a*⊆*c'*) dans la forme normale	$\|C\|^{2}$
Ne conserver que les classes les plus spécifiques	$\|C\|^{2}$

Parmi ces classes, certaines assertions ($o' \in c'$) sont peut-être encore déductibles malgré la suppression de ($o.a=o'$); pour toutes les instances o'' référençant o', récupérer les classes c'' telles que ($o''.a'=o'$), ($o'' \in c$) et ($c.a' \subseteq c''$) dans la forme normale ($o'' \neq o$) $|O|*|A|*|C|^2$

Dans B'_+, on mets l'assertion ($o' \in c'$) si c' n'admet pas de sous-classe parmi les classes c'' $|C|^3$

soit ($|C|^2+|C|^2+|O|*|A|*|C|^2+|C|^3$)=$|\mathbf{C}|^{\mathbf{2}}*(|\mathbf{O}|*|\mathbf{A}|+|\mathbf{C}|)$

(2) Agrandissement d'une restriction d'attribut

Si l'attribut est restreint par un domaine, la complexité est inchangée
Si l'attribut est restreint par une classe :

calculer PCP-1 $|C|^2$

calculer PCP-2 $|C|^2$

agrandir les restrictions d'attribut dans les super-classes $|C|$

ajouter les ($o.a=v$) nécessaires (T4 et T2) $|O|*|A|$

soit $\mathbf{o}(|\mathbf{C}|^{\mathbf{2}}+|\mathbf{O}|*|\mathbf{A}|)$

(3) Résoudre les inconsistances dues à ($o.a=o'$), ($c.a \subseteq c'$) et $\neg(o' \in c')$

Trouver l'ensemble I $|O|*|A|*|C|^2$

(au pire on explore chaque couple (o,a) que l'on marque et on obtient les classes auxquelles doit appartenir l'instance, classes qui sont comparées avec la classe à laquelle on ne veut pas qu'elle appartienne)

S-Référence1(o,a,o',c)

S-réf1-o(o') $|O|*|A|$

(et marquage des objets résolus)

S-réf2-c(c,a) $|A|*|C|$

coût de S2 $|C|^2+|O|*|A|$

rajout des ($o.a=o'$) supprimés inutilement $|O|$

(propagation des suppressions de $ci.ai \subseteq c'i$ aux objets marqués qui étaient problématiques pour ces classes et rajout de ($o.a=o'$) s'il ne reste plus de classe problématique pour o')

soit $\mathbf{o}(|\mathbf{A}|*|\mathbf{C}|*(|\mathbf{O}|*|\mathbf{C}|+|\mathbf{C}|^{\mathbf{2}}+|\mathbf{O}|*|\mathbf{A}|))$

nombre de solutions : $2^{|O|*|A|} + 2^{|C|*|A|}$

(4) Déplacement d'une instance dans une super-classe

S-Référence2 a le même coût que S-Référence1 car la seule différence entre ces deux algorithmes est que S-Référence1 explore les instances référencées et S-Référence2 les instances qui sont référencées.

supprimer ($o \in c$) cste

ajouter ($o \in c'$) si non déductible (T4 et T3) $|C|^2$

ajouter les assertions ($o.a=v$) nécessaires :

récupérer les domaines qui restreignent les classes d'appartenance $|C|$

faire l'intersection de ces domaines et des domaines des classes n'étant pas super-classe de c et sous-classe de c' $|C|$

ajouter les assertions ($o' \in ca$) nécessaires :

pour tous les ($o' \in ca$) qui étaient déductibles (T1 et T2) $|A|*|C|$

non déductible par ($o \in c'$) (T2 et T3) $|C|^2$

ni par une instance qui référence o' $|O|*|A|*|C|^3$

soit $o(|A|*|C|*(|C|^2+|O|*|A|)+|C|^2+|C|^2+|A|*|C|*|O|*|A|*|C|^3)$

$=\mathbf{o(|O|*|A|^2*|C|^4)}$

nombre de solutions : $2^{|O|*|A|} + 2^{|C|*|A|}$

(5) Déplacement d'une classe sous une super-classe

DC :

mise à jour de la table T3 $|C|^2$

héritage des restrictions d'attributs $|A|*|C|$

calcul des ensembles $\{o.a=v\}$ pour $a \in \{a\}$ $|A|*|C|$

Problèmes avec les références

Obtenir Réf>(a) et Réf<(a) pour tous les a en ayant supprimé ceux qui sont inutiles $|A|*|C|^2$

1. Choisir les classes qui sont déplacées $|C|$

pour chacune vérifier qu'elle peut être déplacée $|A|*|C|^2$

déplacer chaque classe sous sa super-classe la plus spécifique qui est aussi déplacée $|C|^2$

modifier leur référence cste

2. Pour chaque classe de Rèf<, $|C|$

modifier sa référence s'il existe au moins une de ses super-classes qui n'a pas été déplacée $|C|^2$

nombre de solutions $2^{|C|}$

Problèmes d'instances :

trouver les instances qu'il faut déplacer (T4) $|O|*|C|$

(celle qui possède une classe d'appartenance ayant été déplacée)

déplace ces instances $|O|*|O|*|A|^2*|C|^4$

soit $o(|C|^2+|A|*|C|+|A|*|C|^2+|C|*|A|*|C|^2*|C|^2+|C|^3$

$+|O|*|C|+|O|*|O|*|A|^2*|C|^4)$

$\mathbf{o(DC)=o(|A|*|C|^4+(|C|+|O|^2*|A))}$

(6) Récapitulatif

modification	complexité
S($o.a=v$)	cste
S($o.a=o'$)	$\|C\|^2*(\|O\|*\|A\|+\|C\|)$
ARA	$\|C\|$
ARAC	$\|C\|^2+\|O\|*\|A\|$
S-réf1-o	$\|O\|*\|A\|$
S-réf1-c	$\|A\|*\|C\|*(\|C\|^2+\|O\|*\|A\|)$
S-Référence1	$\|A\|*\|C\|*(\|O\|*\|C\|+\|C\|^2+\|O\|*\|A\|)$
Déplacement d'une instance	$\|O\|*\|A\|^2*\|C\|^4$
déplacement d'une classe	$\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A\|)$

b) Solutions globales

Dans la colonne intitulée nombre de possibilités, on indique le nombre de possibilités d'application de la solution en question. Par défaut, il y en a une.

$\delta=(o.a=v)$		nb possibilités
solution 1	cste	
solution 2	$\|C\|$	
solution 3	$\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A\|)$	$\|C\|$
solution 4	$\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A\|)$	$\|C\|$
solution 5	$\|O\|*\|A\|^2*\|C\|^4$	$\|C\|$
obtenir toutes les bases	$\|O\|*\|A\|^2*\|C\|^5$	$\|C\|$
obtenir une base	$\|O\|*\|A\|^2*\|C\|^4$	

Il y a un seul choix de l'utilisateur entre 2, 3, 4 et 5.

$\delta=(o.a=o')$		nb possibilités
solution 1	$\|C\|^2*(\|O\|*\|A\|+\|C\|)$	
solution 2	$\|O\|*\|A\|^2*\|C\|^4$	
solution 3	$\|A\|*\|C\|*(\|O\|*\|C\|+\|C\|^2+\|O\|*\|A\|)$	
obtenir toutes les bases	$\|O\|*\|A\|^2*\|C\|^4*2^{\|C\|}$	$2^{\|C\|}$
obtenir une base	$\|O\|*\|A\|^2*\|C\|^4$	

Il y a un choix entre les solutions 2 et 3 pour chacune des classes d'appartenance de l'instance.

$\delta=(o \in c)$		nb possibilités
solution 1'	$\|C\|*\|A\|*\|O\|$	$\|C\|*\|A\|*\|C\|*\|A\|$
solution 2	$\|C\|^2*(\|O\|*\|A\|+\|C\|)$	$2^{\|O\|*\|A\|}$
solution 3	$\|O\|*\|A\|$	
solution 4	$\|A\|*\|C\|*(\|C\|^2+\|O\|*\|A\|)$	
solution 5	$\|A\|*\|C\|*(\|C\|^2+\|O\|*\|A\|)$	
solution 6	$\|O\|*\|A\|^2*\|C\|^4 * \|C\|*\|A\|$	$2^{\|C\|}$
obtenir toutes les bases	$\|O\|*\|A\|*\|C\|^2*(\|O\|*\|A\|+\|C\|)*2^{\|O\|*\|A\|}$ $+\|O\|^2*\|A\|^4*\|C\|^5*2^{\|C\|}$	$\|C\|^2*\|A\|^2*2^{\|O\|*\|A\|}*$ $2^{\|C\|}$
obtenir une base	$\|O\|^2*\|A\|^3*\|C\|^5$	

$\delta=(c.a \subseteq d)$		nb possibilités (une par défaut)
PCI et PCP	$\|C\|$	
solution 1	$\|C\|$	
solution 2	$\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A)$	
solution 3	$(\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A\|))*\|C\|$	
solution 4	$\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A\|)$	
solution 5	$\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A\|)$	
solution 6	cste	
solution 7	$\|C\|^2*(\|O\|*\|A\|+\|C\|)$	
solution 7'	$\|C\|^2*(\|O\|*\|A\|+\|C\|)$	
solution 8	$\|O\|*\|A\|^2*\|C\|^4$	
obtenir toutes les bases	$\|A\|*\|C\|^5+$ $(\|A\|*\|C\|^4+(\|C\|+\|O\|^2*\|A\|))*2^{\|C\|}$ $+\|O\|*\|A\|^2*\|C\|^4*2^{\|I\|}$	
obtenir une base	$\|O\|^2*\|A\|^2*\|C\|^5$	

Il y a un choix de l'utilisateur entre les solutions 1, 2 et 3 pour la classe *c*, un choix entre 4, 5 et 6 pour toutes les sous-classes et un choix entre 7, 7' et 8 pour toutes les instances.

5. Comparaison avec d'autres travaux

a) L'abduction logique

Le raisonnement utilisé pour trouver les bases révisées est similaire à celui utilisé en abduction : choix entre les formules que l'on ne veut pas déduire et pour chacune recherche de la manière de ne plus pouvoir les déduire. Cependant, des différences apparaissent à divers niveaux :

- la structure de la base de connaissance par objets permet d'ordonner les solutions présentées à l'utilisateur qui pourra ainsi mieux comprendre pourquoi il doit choisir entre telle ou telle suppression.
- pour obtenir une solution quelconque, le coût n'est pas exponentiel; la complexité exponentielle n'apparaît que si toutes les bases révisées doivent être obtenues.
- il n'est pas nécessaire de faire de retour arrière sauf avis express de l'utilisateur, car les choix ne conduisent jamais à des impasses (grâce à la minimalité de la forme première qui fait que toute suppression est effective)
- il n'est pas nécessaire de tester la consistance à chaque pas (à chaque suppression), car le test d'acceptabilité permet d'obtenir l'ensemble des inconsistances et incohérences et que à chaque modification de la base, les problèmes résolus sont facilement identifiables.

b) Nebel

[Nebel90] énonce dans l'un de ses postulats sur la complexité de l'opération de révision qu'elle ne doit pas être plus complexe que la déduction. En effet, la révision consiste pour lui à déterminer les conséquences de la base révisée. Nous adoptons ici une stratégie un peu différente car nous souhaitons obtenir explicitement la base révisée sous forme première, ce qui explique que la complexité est exponentielle pour obtenir l'ensemble des bases révisées.

L. Minimalité de l'algorithme

La minimalité des solutions proposées par l'algorithme est montrée en utilisant la proposition 27. Chaque point de la proposition est étudiée pour chaque type d'assertion qui a provoqué l'inconsistance.

Le point (0) est immédiat car les solutions élémentaires permettent de résoudre une inconsistance donnée, qu'une solution élémentaire est appliquée pour chaque inconsistance sauf si une solution élémentaire permet de résoudre plusieurs inconsistances et surtout que les ajouts faits par les solutions élémentaires ne soulèvent pas d'autres inconsistances.

Le point (1) : si $B''_{-} \subseteq B'_{-}$ et $B'_{+} \subseteq B''_{+}$, alors $B''=B'$.

est montré en deux étapes.

- Tout d'abord, une base dans laquelle un sous-ensemble des suppressions faites dans les solutions proposées sont effectuées ne peut être consistante.
- En second lieu, on montre que toutes les assertions de la clôture déductive de la base initiale qui ne sont plus déductibles de la base révisée proposée par l'algorithme ne sont pas acceptables. On utilise pour cela l'inclusion suivante : $B''_{+} \subseteq \mathrm{Cn}(B) - \mathrm{Cn}(\mathrm{FP}(B) - B'_{-}) - B'_{-}$, ce qui se démontre trivialement par l'écriture de base révisée utilisant B'_{+} et B'_{-}.Si $B'_{+} \subseteq B''_{+}$, il

faut montrer que pour tout δ tel que δ est dans cet ensemble, soit $B' \vdash \delta$, soit elle est inacceptable. En effet, si $B' \vdash \delta$, $\mathrm{Cn}(B') \subseteq \mathrm{Cn}(B'')$ et donc $\mathrm{Cn}(B')=\mathrm{Cn}(B'')$. De plus, comme $B''_+ \subseteq \mathrm{Cn}(B) - \mathrm{Cn}(\mathrm{FP}(B) - B'_-) - B'_-$ et que B'_- est une union d'ensembles qui correspondent chacun à la résolution d'une inconsistance. Ces ensembles peuvent être regroupés selon la solution élémentaire adoptée. Les problèmes soulevés par l'ajout d'une assertion seront numérotés et Ij indiquera l'ensemble des problèmes résolus par la solution j. Si on décompose l'inclusion précédente selon toutes les solutions élémentaires utilisées en appelant B'_{-i}, les suppressions associées à la résolution de l'inconsistance *i*, on obtient :

$$B''_+ \subseteq \bigcup_{j \in \{\text{solution élémentaire}\}} \bigcup_{i \in I_j} \{\mathrm{Cn}(B) - \mathrm{Cn}(\mathrm{FP}(B) - B'_{-i}) - B'_{-i}\}$$

Il s'agit donc de traiter les assertions de chaque ensemble $\mathrm{Cn}(B) - \mathrm{Cn}(\mathrm{FP}(B) - B'_{-i}) - B'_{-i}$, noté *Suppr(i)*.

Pour les assertions du type ($o.a=v$), on étudie chaque modification élémentaire séparément, pour montrer que pour chaque suppression d'assertion, les assertions du type ($o.a=v$) ont été ajoutées. Cette assertion est étudiée séparément car son ajout ne dépend que de la solution élémentaire.

Les points suivants, (2), (3), (4), (5), (6) et (7), sont montrés en étudiant les assertions de B'_+.

1. Les solutions élémentaires

Pour chaque solution élémentaire, nous allons montrer que les assertions qui ne peuvent plus être déductibles suite aux retraits des assertions de B'_-, correspondent aux assertions qui ont été ajoutées dans B'_+.

Nous définissons ci-dessous des ensembles d'assertions qui vont permettre de décrire les assertions qui sont supprimées par la suppression des assertions de B'_-. Pour chaque solution élémentaire, on notera *suppr* l'ensemble des assertions supprimées de la clôture déductive, autres que les assertions de B'_-, suite à la suppression de B'_-.

DÉFINITION : ENSEMBLES D'ASSERTIONS

E1(c_n,c_m,a)=$\{(c_i.a \subseteq d'_i)$ t.q. $d'_i \in T$, $d'_i > : d_i$, $(c_n \leq^* c_i \leq^* c_{m-1}) \in \mathrm{FP}(B)$ et $(c_i.a \subseteq d_i) \in \mathrm{FN}(B)\}$
les restrictions de l'attribut a pour l'ensemble de classes comprises entre c_n et c_m.

E1'(c_n,c_m,a)=$\{(c_i.a \subseteq d'_i)$ t.q. $d'_i \in C$, $(d_i \leq^* d'_i) \in \mathrm{FP}(B)$, $(c_n \leq^* c_i \leq^* c_{m-1}) \in \mathrm{FP}(B)$ et $(c_i.a \subseteq d_i) \in \mathrm{FN}(B)\}$
identique à la précédente mais pour les restrictions par des classes.

E2(c_n,c_m,a)=$\{(c_j.a \subseteq d'_i)$ t.q. $\exists i$, $d'_i \in T$, $d'_i \geq : d_i$, $(c_j \neq c_i)$, $(c_j \leq^* c_i) \in \mathrm{FP}(B)$, $(c_n \leq^* c_i \leq^* c_{m-1}) \in \mathrm{FP}(B)$ t.q. $(c_i.a \subseteq d_i) \in \mathrm{FN}(B)$ et $(c_j.a \subseteq d_i) \in \mathrm{FN}(B)\}$

les restrictions déductibles dans les sous-classes d'un ensemble de classes comprises entre c_n et c_m.

E2'(c_n,c_m,a)={($c_j.a \subseteq d'_i$) t.q. $\exists i$ $d'_i \in C$, ($c_j \neq c_i$), ($c_j \leq c_i$)$\in$ Cn(B), ($c_n \leq c_i \leq c_{m-1}$)$\in$ Cn(B) t.q. ($c_i.a \subseteq d_i$)$\in$ FN(B), ($d_i \leq^* d'_i$)$\in$ FP(B) et ($c_j.a \subseteq d_i$)$\in$ FN(B)}
identique à la précédente mais pour des restrictions par des classes.

E3(c,a)={$o' \in c'_i$ t.q. $\exists i$ ($c \leq^* c_i \leq^* c_{m-1}$)$\in$ FP(B), ($o \in c_i$)$\in$ Cn(B), ($o.a=o'$)$\in$ FP(B), ($c_i.a \subseteq c''_i$)$\in$ FP(B), ($c''_i \leq^* c'_i$)$\in$ FP(B)}
les rattachements d'instances déductibles par (o-valuation)

E4(c)={$c_k \leq c_j$, ($c_k \leq c$)$\in$ Cn(B) et ($c' \leq c_j$)$\in$ Cn(B) avec ($c \leq c'$)$\in$ FP(B)}\{$c \leq c_{i+1}$}
les relations de spécialisation entre classes déductibles grâce au lien de spécialisation entre c et sa super-classe.

E5(c)={$o' \in c_j$, ($c_i \leq^* c_j$)$\in$ FP(B), $\exists k$ ($c_k \leq c$)$\in$ Cn(B), ($o' \in c_k$)$\in$ FP(B) et $\neg\exists l$ ($c_l \neq c_k$) ($c_l \leq^* c_j$)$\in$ FP(B) et ($o' \in c_l$)$\in$ FP(B)}
les rattachements d'instances déductibles grâce au lien de spécialisation entre c et sa super-classe.

E6(o,c)={$o \in c_j$, ($c \leq^* c_j$)$\in$ FP(B) et $\neg\exists l$ ($c_l \neq c_k$) ($c_l \leq c_j$)$\in$ Cn(B) t.q. ($o' \in c_l$)$\in$ FP(B)}
les rattachements de l'instance o déductibles grâce à l'attachement de o à c.

E7(o,c)={$o' \in c_a$ t.q. $\exists$ a tel que ($o.a=o'$)$\in$ FP(B), ($c.a \subseteq^* c_a'$)$\in$ FP(B),
$\neg\exists$ c_a'' telle que ($c_a'' \leq^* c_a'$)$\in$ FP(B) et ($c'.a \subseteq^* c_a''$)$\in$ FP(B)
et $\neg\exists$ o'', c'', a'', c_a'' tels que ($c''.a'' \subseteq^* c_a''$)$\in$ FP(B), ($o''.a''=o'$)$\in$ FP(B) et ($c_a'' \leq^* c_a'$)$\in$ FP(B)}

E8(o,c)={$o' \in c'$ s'il existe c telle que ($o \in c$)$\in$ Cn(B), ($c.a \subseteq^* c'$)$\in$ FP(B)

il n'existe pas $c'' \neq c'$ et $c''' \neq c$ telles que ($o \in c'''$)$\in$ Cn(B), ($c'' \leq^* c'$)$\in$ FP(B) et ($c'''.a \subseteq^* c''$)$\in$ FP(B),

et il n'existe pas o'', c'', a'', c''' tels que ($o \in c''$)$\in$ FN(B), ($c''.a'' \subseteq^* c'''$)$\in$ FP(B), ($o''.a''=o'$)$\in$ FN(B) et ($c''' \leq^* c'$)$\in$ FP(B)}}

E9(c,a,d)={$o.a=v$ tel que {v}=$\bigwedge${d_j t.q. ($o \in c_j$)$\in$ FN(B) et ($c_j.a \subseteq d_j$)$\in$ FN(B)}

et {v}$\neq\bigwedge${d_j t.q. ($o \in c_j$)$\in$ FN(B), ($c_j.a \subseteq d_j$)$\in$ FN(B) et ($c \leq^* c_i \leq^* c_{m-1}$)$\notin$ FP(B)}

$\wedge\bigwedge${$d_j \vee d$ t.q. ($o \in c_j$)$\in$ FN(B), ($c_j.a \subseteq d_j$)$\in$ FN(B) et ($c \leq^* c_i \leq^* c_{m-1}$)$\in$ FP(B)}

a) Suppression d'une valeur d'un attribut d'une instance

S(o,a,v) : $B'_- = \{o.a=v\}$;

(1) *suppr*=Ø ne supprime aucune assertion (autre que l'assertion effectivement supprimée).

Aucun autre point n'est concerné.

b) Suppression de la référence à une instance par un attribut d'une instance

S(o,a,o') : $B'_{-}=\{o.a=o'\}$ entraîne la suppression d'assertions du type ($o' \in c$) et donc du type ($o'.a=v$).

(1)*suppr*=E8(o,c)$\cup\{o'.a'=v'$ tel que cette assertion n'est déductible qu'en utilisant une assertion de E8(o,c)$\}$

Cependant, E8(o,c) est ajouté dans B'_{+}, ce qui permet également de déduire les assertions du second ensemble.

(3) soit ($o' \in c'$)$\in B'_{+}$; soit ($c'' \leq^* c$)$\in$ FP(B), l'assertion ($o' \in c''$) est soit telle que $c'' \in$ CNC et donc elle est inacceptable, soit elle n'appartient pas à la clôture déductive de la base car elle n'appartient pas à la forme première et qu'elle ne peut être déductible par (o-valuation)

(6) soit ($o' \in c'$)$\in B'_{+}$; E-réf(o',c')=$\{(o,a)\}$ et ($o.a=o'$) est inacceptable dans la base.

c) Agrandir une restriction d'attribut

Attribut restreint par un domaine : ARA (c,a,d)

Certaines restrictions d'attributs de classe sont élargies donc leurs instances ont pu perdre leur valeur d'attribut, si celle-ci était déduite par la règle de (o-valuation). Ce sont les instances telles que l'intersection des restrictions de domaine de leur classes d'appartenance est égal à une valeur et dont l'intersection des restrictions après l'agrandissement des domaines n'est plus égale à cette valeur.

D'autre part, toutes les restrictions d'attributs pour l'attribut concerné dans les sous-classes des classes dont le domaine est agrandi sont perdues. L'ajout de d en entier dans les domaines des super-classes de la classe problématique n'était pas nécessaire (cf. la remarque à ce propos dans le paragraphe concernant l'agrandissement de restriction). Pour toutes les classes qui ne sont pas super-classes de la classe problématique et dont l'une des super-classes directes a une restriction d'attribut modifiée, si sa restriction d'attribut correspond à la restriction de sa super-classe, cette restriction est perdue et est donc ajoutée dans B'_{+}. Les assertions du type ($o \in c$) et ($c \leq c'$) ne peuvent être touchées par des suppressions de restrictions de domaines.

Soient c_n=PCI(c,a,d) et c_m=PCP($c,a,\{v\}$)

(1) *suppr(ARA(c,a,d))*=E1(c_n,c_m,a)$\cup$ E2(c_n,c_m,a)$\cup$ E9(c,a,d)

Soit ($c_i.a \subseteq d'_i$)$\in$ E1(c_n,c_m,a); ($c_i.a \subseteq d_i \cup d$)$\in B'_{+}$, donc $B' \vdash (c_i.a \subseteq d'_i)$ $\forall d'_i \geq: d_i \cup d$ $\forall i$ ($c_n \leq^* c_i \leq^* c_{m-1}$)$\in$ FP(B) si ($c_i.a \subseteq d_i$)$\in$ FP(B). Pour les assertions ($c_i.a \subseteq d'_i$) telles que

$d'_i{>}{:}d_i$ et $\neg d'_i{\geq}{:}d_i \cup d$, nous nous sommes fixés de ne pas les insérer, même si elles peuvent être acceptables; c'est une entorse faite au principe de minimalité.

Soit $(c_j.a{\subseteq}d'_i){\in}\,\mathrm{E2}(c_n,c_m,a)$; $\exists i$ $(c_j{\neq}c_i)$, $(c_j{\leq}c_i){\in}\,\mathrm{Cn}(B)$, $(c_n{\leq}c_i{\leq}c_{m-1}){\in}\,\mathrm{Cn}(B)$ t.q. $(c_i.a{\subseteq}d_i){\in}\,\mathrm{FN}(B)$ $d'_i{\geq}{:}d_i$; soit $c_{j'}$ telle que $(c_{j'}{\leq}c_i){\in}\,\mathrm{FP}(B)$, $(c_j'.a{\subseteq}d_i){\in}\,B'_+$, donc $B'{\vdash}(c_j.a{\subseteq}d'_i)$.

$\mathrm{E9}(c,a,d){\subseteq}\,B'_+$.

(4) $(c_i.a{\subseteq}d_i{\cup}d){\in}\,B'_+$; ce cas est une transgression à la minimalité car les assertions de la forme $(c_i.a{\subseteq}d_i{\cup}\{v\})$ avec $v{:}d$ sont acceptables, mais nous avons vu qu'il est préférable que les nouveaux domaines contiennent le domaine d tout entier plutôt qu'une valeur arbitraire.

$(c_j.a{\subseteq}d_i){\in}\,B'_+$; $(c_j.a{\subseteq}d_i){\in}\,\mathrm{FN}(B)$ donc il n'existe pas d'_i ${<}{:}d_i$ t.q. $(c_j.a{\subseteq}d'_i){\in}\,\mathrm{Cn}(B)$; de plus, $(c_i.a{\subseteq}d_i){\in}\,\mathrm{FN}(B)$, donc il n'existe pas $(c_j{\leq}{*}c'){\in}\,\mathrm{FP}(B)$ et d'_i ${<}{:}d_i$ t.q. $(c'.a{\subseteq}d'_i){\in}\,\mathrm{Cn}(B)$.

(7) soit $(o.a{=}v){\in}\,B'_+$; $\{v\}{=}\mathrm{Rest}(o,a,\emptyset)$ et il existe au moins une assertion $(c_i.a{\subseteq}d_i)$ qui est inacceptable.

Attribut restreint par une classe : ARAC(c,a,c')

La modification des classes référencées par des super-classes entraîne la perte d'assertions du type $(o'{\in}\,c)$ et donc du type $(o'.a{=}v)$ et également $(c.a{\subseteq}c')$ pour les sous-classes des classes dont les restrictions sont modifiées. Cependant, les assertions du type $(o'{\in}\,c)$ qui auraient été perdues sont ajoutées dans B'_+, donc aucune assertion du type $(o.a{=}v')$ n'est supprimée; elles ne seront donc pas mentionnées dans l'ensemble *suppr* ci-dessous. D'autre part, les assertions du type $(c{\leq}c')$ ne peuvent être touchées par des modifications de restrictions d'attribut de classes.

(1) $\mathit{suppr}(\mathit{ARAC}(c,a,c')){=}\mathrm{E1}'(c,c_m,a){\cup}\mathrm{E2}'(c,c_m,a){\cup}\mathrm{E3}(c,a)$

Soit $(c_i.a{\subseteq}d'_i){\in}\,\mathrm{E1}'(c,c_m,a)$; $(c_i.a{\subseteq}d'_i)$ avec $\neg(c'_{m-1}{\leq}{*}d'_i){\in}\,\mathrm{FP}(B)$ n'est pas acceptable; soit $(c_i.a{\subseteq}d'_i)$ avec $(c'_{m-1}{\leq}{*}d'_i){\in}\,\mathrm{FP}(B)$, $(c_{m-1}.a{\subseteq}\mathit{PSSCC}(c'_{m-1},c')){\in}\,B'_+$; $(d_i{\leq}{*}d'_i){\in}\,\mathrm{FP}(B)$, $(c{\leq}{*}c_i{\leq}{*}c_{m-1}){\in}\,\mathrm{FP}(B)$, $(c_i.a{\subseteq}d_i){\in}\,\mathrm{FN}(B)$ et $(c.a{\subseteq}c'){\in}\,\mathrm{FP}(B)$ donc $(c'{\leq}{*}d'_i){\in}\,\mathrm{FP}(B)$ et comme de plus $(c'_{m-1}{\leq}{*}d'_i){\in}\,\mathrm{FP}(B)$, $(\mathit{PSSCC}(c'_{m-1},c'){\leq}{*}d'_i){\in}\,\mathrm{FP}(B)$; puisque $(c_i{\leq}{*}c_{m-1}){\in}\,\mathrm{FP}(B)$, on a alors $B'{\vdash}(c_i.a{\subseteq}d'_i)$.

Soit $(c_j.a{\subseteq}d'_i){\in}\,\mathrm{E2}'(c,c_m,a)$; ce cas est similaire à la démonstration de l'appartenance des assertions de E2 pour ARA.

Soit $(o'{\in}\,c'_i\,){\in}\,\mathrm{E3}(c,a)$; $(o'{\in}\,c''_i\,){\in}\,B'_+$ avec $(c''_i{\leq}{*}c'_i){\in}\,\mathrm{FP}(B)$, donc $B'{\vdash}(o'{\in}\,c'_i\,)$

(3) $(o' \in c'_i) \in B'_+$; il n'existe pas $(c' \leq^* c'_i) \in \text{FP}(B)$ telle que $(o' \in c') \in \text{Cn}(B)$, car $(o' \in c') \notin \text{FP}(B)$ et E-réf$(o',c')=\emptyset$.

(6) $(o' \in c'_i) \in B'_+$; pour tout $(o,a) \in$ E-réf(o',c'_i), $(o \in c_i) \in \text{Cn}(B)$ et $(c \leq^* c_i \leq^* c_{m-1}) \in \text{FP}(B)$ et $(c_i.a \subseteq d'_i)$ est inacceptable.

(5) $(c_{m-1}.a \subseteq c'') \in B'_+$; c''=PSSCC(c',c'_{m-1}), donc $(c_{m-1}.a \subseteq c''')$ avec $(c''' \leq^* c'') \in \text{FP}(B)$ est inacceptable, de même donc pour les sous-classes de c_{m-1}.

$(c_j.a \subseteq c'_i) \in B'_+$; cette démonstration est similaire à celle donnée dans ARA-(4).

d) Résoudre les inconsistances soulevées par la présence de $o.a=o'$, $c.a \subseteq c'$ et $(o' \in c')$ inacceptable

S-Référence1(o,a,o',c)

Cette solution combine différentes solutions élémentaires; il s'agit de vérifier que l'on ne fait pas trop de suppressions (point (1) de la proposition P%27). Par contre, les autres points ne concernent que les ajouts effectués dans B'_+ et comme le résultat est obtenu pour chaque modification élémentaire, il est obtenu pour cette modification qui les combinent.

(1) il faut vérifier que chaque modification élémentaire effectuée est nécessaire.

$(o'.a_i=o_i)$ est supprimée uniquement si a_i est un attribut problématique, c'est-à-dire que l'assertion apparaît dans une inconsistance qui n'a pas été résolue par une coupure de la chaîne en amont de l'attribut a_i. Les suppressions d'attributs dans la suite de la chaîne ne sont pas proposées. D'autre part, des modifications des restrictions d'attribut peuvent rendre à nouveau acceptable des assertions $(o'.a_i=o_i)$, mais dans ce cas, les assertions sont ré-ajoutées.

$(c.a \subseteq c')$ est supprimée uniquement s'il reste des inconsistances concernant la restriction par la classe c'. De plus, les suppressions dans le reste de la chaîne ne sont pas proposées si la suppression a été choisie par l'utilisateur.

e) Déplacer une instance dans une super-classe

DI(o,c,c')

Le déplacement entraîne la perte d'une valeur d'un attribut de l'instance déplacée si l'intersection des domaines de ses classes d'appartenance avant le déplacement était une valeur et si cette intersection en remplaçant la nouvelle classe d'appartenance par l'ancienne n'est plus égale à cette valeur. Les assertions de ce type sont ajoutées dans B'_+.

D'autres instances perdent leur valeur si le rattachement à certaines de leurs classes d'appartenance est perdu; or, tous les rattachements qui auraient été perdus sont ajoutés dans B'_+.

(1) *suppr*(DI(o,c,c'))= E6(o,c)$\cup$E7(o,c)

Soit ($o\in c_i$)$\in$E6(o,c); ($o\in c'$)$\in B'_+$ donc $B'\vdash o\in c_i$ si ($c'\leq c_i$)$\in$Cn(B). De plus, ($o\in c_i$) avec i tel que ($c_i\leq c'$)$\in$Cn(B), ($c_i\neq c'$) est inacceptable d'après la pré-condition du déplacement d'instance

E7(o,c)$\subseteq B'_+$.

(3) ($o\in c'$)$\in B'_+$; d'après la pré-condition, le rattachement de l'instance à une sous-classe de c' qui soit également super-classe de c est inacceptable. D'autre part, s'il existe une sous-classe de c' qui est classe d'appartenance de o, l'assertion n'est pas ajoutée dans B'_+; enfin, s'il n'existe pas de sous-classe de c qui soit classe d'appartenance de o, une telle assertion n'appartient pas à la clôture déductive de la base.

($o'\in c_a$)$\in B'_+$; il n'existe pas ($c'\leq^* c_a$)$\in$FP(B) telle que ($o'\in c'$)$\in$Cn(B), car ($o'\in c'$)$\notin$FP(B) et E-réf(o',c')=Ø.

(6) ($o\in c'$)$\in B'_+$; d'après la définition, E-réf(o,c')=Ø.

($o'\in c_a$)$\in B'_+$; pour tout (o,a)$\in$E-réf(o',c_a), ($o\in c$) est inacceptable, avec ($c.a\subseteq c_a$)$\in$FN(B).

(7) soit ($o.a=v$)$\in B'_+$; {v}=Rest(o,a,Ø) et ($o\in c$) est inacceptable.

Les assertions ajoutées par S-Référence2 sont traitées dans le paragraphe suivant consacré à cette modification élémentaire.

f) S-référence2 : les instance qui référencent une instance déplacée

De même que S-référence1, cette modification se compose de modifications élémentaires; donc seul le point (1) est à démontrer.

(1) S(o',a_i,o,CNC) n'est effectué à un moment de l'algorithme ou il existe des inconsistances faisant intervenir ($o'.a_i$); de plus si l'une des solutions du type ARAC permet de lever des inconsistances précédemment résolues par S(o',a_i,o,CNC), ces suppressions sont supprimées. De plus, il faut noter que l'ensemble CNC contient les classes d'appartenance problématique pour l'instance et seulement celles-là.

ARAC(c_i,a_i,c'_a) n'est effectué que s'il reste des inconsistances faisant intervenir l'attribut a_i et aucune modification n'est effectuée dans la suite de la chaîne.

g) Déplacer une classe sous une super-classe

DC(c,c',ANC)

L'ensemble *ANC* est l'ensemble des attributs non conservés lors du déplacement. Pour chacun de ces attributs, les instances de la classe déplacée et de ses sous-classes ont pu perdre leur valeur, mais l'assertion ($o.a=v$) est ajoutée dans ce cas dans B'_+.

Pour les attributs a' n'appartenant pas à ANC, les restrictions sur la classe sont conservées donc les assertions du type ($o.a'=v$) ne sont pas touchées.

Soit *racine* la classe telle que ($c \leq^* racine$) $\in$ FP(B) et il n'existe pas c' telle que ($racine \leq c'$) $\in$ FP(B).

(1) $suppr(\mathrm{DC}(c,c',ANC)) = \{\mathrm{E2}(c,c,a)\}_{a \in \mathrm{A}} \cup \{\mathrm{E2'}(c,c,a)\}_{a \in \mathrm{A}} \cup \mathrm{E4}(c) \cup \mathrm{E5}(c)$

Soit ($cj.a \subseteq d'$) $\in$ E2(c,c,a); ($c_j \neq c_i$), ($c_j \leq^* c$) $\in$ FP(B) t.q. ($c.a \subseteq d$) $\in$ FN(B), $d' \geq: d$ et ($c_j.a \subseteq d_i$) $\in$ FN(B); or, si $a \notin ANC$, ($c.a \subseteq d$) $\in B'_+$, d'où $B' \vdash cj.a \subseteq d'$. Si $a \in ANC$, ($c.a \subseteq d$) est inacceptable dans la base révisée; quant aux restrictions dans les sous-classes, il faut étudier les solutions spécifiques à chaque type d'assertion ajoutée : soient elles sont inacceptables, soit elles sont ajoutées dans une solution particulière.

Pour ($cj.a \subseteq d'$) $\in$ E2'(c,c,a), le raisonnement est identique.

Soit ($c_k \leq c_i$) $\in$ E4(c); ($c_i \leq c'$) $\in B'_+$, donc $B' \vdash c_k \leq c_j$ si ($c' \leq c_j$) $\in$ Cn(B) et ($c_k \leq c_i$) $\in$ Cn(B). De plus, $c_k \leq c_j$ avec ($c_{i+1} \leq c_j$) $\in$ Cn(B) et ($c_j \leq^* c'$) $\in$ FP(B) est inacceptable selon la pré-condition posée sur c'.

E5(c) $\subseteq B'_+$.

(2) ($c_i \leq c_m$) $\in B'_+$; d'après la pré-condition, ($c_i \leq c'$) avec ($c' \leq^* c_m$) $\in$ FP(B) et $c' \neq c_m$ est inacceptable.

(4) ($c_i.b \subseteq d_j$) $\in B'_+$; soit ($c'_i.b \subseteq d'_j$) avec $B' \vdash (c_i \leq^* c'_i)$, $d'_i \leq: d_i$ et (c'_i,d'_i) $\neq$ (c'_i,d'_i); $d'_i = d_i$ sinon l'assertion n'appartient pas à Cn(B); d'autre part, l'assertion n'est ajoutée que si ($c_m.b \subseteq d_j$) n'appartient pas à la forme normale, donc que l'assertion ($c'_i.b \subseteq d'_j$) n'appartient pas à Cn(B).

(7) soit ($o.a=v$) $\in B'_+$; Rest($o,a,\emptyset$)=$\{v\}$, mais il existe des assertions ($c_j.a \subseteq d_j$) inacceptables, avec ($c_i \leq^* c_j \leq^* c_m$) $\in$ FP(B) (car elles sont la cause du déplacement).

Toutes les solutions qui se composent de solutions élémentaires vérifient également (1), si les pré-conditions des solutions élémentaires sont remplies. En effet, pour les ensembles de suppressions correspondant aux solutions élémentaires, toutes les assertions qui ne seraient plus déductibles sont ajoutées

2. Les solutions globales

Pour chaque type d'assertion, il ne reste à présent qu'à montrer la première partie du point (1) de la proposition, à savoir que toutes la suppression de toutes les assertions de B'_- sont nécessaires pour obtenir une base consistante : si $B''_- \subseteq B'_-$ alors $B''_- = B'_-$, sinon B'' n'est pas consistante.

a) Ajout de l'assertion $o.a=v$

Si $(o.a=v') \in B'_-$, toute base contenant $(o.a=v)$ et $(o.a=v')$ avec $v \neq v'$ est inconsistante.

Les différentes modifications élémentaires sont appliquées pour chaque classe d'appartenance de o qui est impliquée dans une inconsistance; les diverses modifications proposées n'interfèrent pas car elles ne résolvent toutes que l'inconsistance pour la classe en question ; notons cependant que ceci n'est vrai que si on a posé comme hypothèse qu'il n'y a qu'une seule classe d'attachement par point de vue.

b) Ajout de l'assertion $o.a=o'$

si $(o.a=o'') \in B'_-$, toute base contenant $(o.a=o'')$ et $(o.a=o')$ avec $o \neq o'$ est inconsistante.

Pour les assertions du type $(o'.a'=o'')$ supprimer par S-Référence1, l'algorithme vérifie après la modification des restrictions d'attribut que ces suppressions sont toujours utiles.

Les suppressions d'attribut sont effectuées pour résoudre des inconsistances qui n'ont pas été résolues par d'autres suppressions précédentes, donc elles sont nécessaires.

Si des déplacements d'instances sont devenus superflus, ils sont supprimés à la fin de l'algorithme

De plus, de même que pour $(o.a=v)$, les modifications élémentaires n'interfèrent pas.

c) Ajout de l'assertion $o \in c$

Il s'agit de vérifier qu'une modification élémentaire n'est pas faite de manière superflue, c'est-à-dire si elle ne pouvait pas être évitée.

Pour résoudre tout d'abord les inconsistances portant directement sur l'instance o.

(S6) résout les inconsistances pour un ensemble d'attributs donnés et S4 et S5 font des modifications pour des attributs différents et des classes référencées par des classes différentes.

Pour les inconsistances portant sur les instances référencées par o :

La solution (S1') qui permet la résolution pour plusieurs attributs est effectuée en premier, ce qui permet de ne proposer les autres solutions que pour les attributs non résolus par cette solution. De plus, S2 et S3 sont appliquées pour des attributs différents et donc toutes les

modifications proposées sont nécessaires car résoudre le problème pour un attribut par l'une de ces deux solutions ne peut résoudre l'inconsistance pour un autre attribut. Enfin, il reste éventuellement des inconsistances de type (3) qui sont alors résolues pour chaque instance inconsistance. De plus, ces solutions ne peuvent résoudre des inconsistances de type (2).

d) Ajout de l'assertion $c.a \subseteq d$

Les modifications élémentaires sont effectuées en premier lieu sur les classes les moins spécifiques. Les modifications sur les classes plus spécifiques ne peuvent résoudre les inconsistances sur leurs super-classes et de plus, les modifications pour deux classes non liées par relation de spécialisation sont indépendantes. Toutes les modifications effectuées sur les classes sont donc nécessaires.

e) Ajout de l'assertion $c.a \subseteq c'$

Si la classe c est incohérente, il faut lever cette incohérence par l'une des solutions proposées : l'une des modification S1, S2, S2' ou S3 est donc nécessaire.

Comme pour les assertions du type ($c.a \subseteq d$), la modification élémentaire effectuée pour rendre cohérente une classe plus spécifique qu'une autre, ne permet de rendre cohérente que cette dernière. L'algorithme propose donc des modifications pour les classes les moins spécifiques; suite à ces modifications, de l'ensemble des classes incohérentes sont retirées les classes qui ont retrouvées leur cohérence par la modification nécessitée par une super-classe.

Pour les instances :

chaque modification résout l'inconsistance pour une instance. Vu la définition de S-réf1-o, il n'y a pas de suppression inutile pour une instance particulière; de plus, à chaque suppression d'une instance de la forme ($o.a=v$), il est vérifié que cela ne rend pas inutile une suppression précédente et cela répercute sur les autres instances le fait que cela est résolu.

Bibliographie

[Abadi&96] M. Abadi, L. Cardelli. A theory of objects. Springer-Verlag. Monographs in computer science. 1996.

[Abiteboul88] S. Abiteboul. *Updates : a new frontier*. LNCS 326. pp.1-18. 1988.

[Alchourrón&85] C. Alchourrón, E. Gärdenfors, P. Makinson. *On the Logic of Theory Change: Partial Meet Contraction and Revision Functions*. Journal of Symbolic Logic. 50(2):510-530. 1985.

[Aravindan&94] C. Aravindan, P. Dung. *Belief dynamic, abduction, and databases*. LNCS (LNAI) 838. pp.66-85. 1994.

[Barnejee&87] J. Barnejee, W. Kim, H. Kim, H. Korth. *Semantics and Implementation of Schema Evolution in Object-oriented Databases*. SIGMOD Records. 16(3):311-322. 1987.

[Barral&92] C. Barral, S. Kraus, J. Minker, V. Subrahmanian. *Combining knowledge bases consisting of first-order theories.* Computational Intelligence. 8(1):45-71. 1992.

[Benferhat&93] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, H. Prade. *Inconsistency management and prioritized syntax-based entailment.* Actes 11ième IJCAI. Chambéry (FR). pp.640-645. 1993.

[Besnard91] P. Besnard. *Paraconsistent logic approach to knowledge representation.* First World Conference on the Fundamentals of Artificial Intelligence. Paris (FR). pp.107-114. 1991.

[Boutilier94] C. Boutilier. *Unifying default reasoning and belief revision in a modal framework*. Artificial Intelligence. 68:33-85. 1994.

[Borgida85] A. Borgida. *Language features for flexible handling of exceptions in informations systems.* ACM Transactions on Database system. 10(4):565-603. 1985.

[Brachman&85] R. Brachman, J. Schmolze. *An overview of the KL-ONE Knowledge Representation System*. Cognitive Science. 9(2):171-216. 1985.

[Brewka89] G. Brewka. *Preferred subtheories : an extended logical framework for default reasoning*. Actes 11ième IJCAI. pp.1043-1048. Détroit (MI US). 1989.

[Capponi95] C. Capponi. *Identification et exploitation des types dans un modèle de connaissances à objets.* Thèse d'informatique. Université Joseph Fourier. Grenoble (FR). 1995.

[Cholvy94] L. Cholvy. *Fusion de sources d'informations ordonnées en fonction des thèmes.* Actes 9ième RFIA. Paris (FR). pp.487-494. 1994.

[Cordier&94] M.O. Cordier, P. Siegel. *A temporal revision model for reasoning about world changs*. Special issue on Revision and Updating in Knowledge Base. Léa Sombé (eds.). International Journal of Intelligent Systems. 9(1):131-142. 1994.

[Crampé&96a] I. Crampé, J. Euzenat. *Révision interactive dans une base de connaissance par objets*. Actes 10ième RFIA. Rennes (FR). pp.615-623. 1996.

[Crampé&96b] I. Crampé, J. Euzenat. *Fondements de la révision dans un langage d'objets simple*. Rapport de recherche INRIA n°3060. 1996.

[Dalal88] M. Dalal. *Investigations into a Theory of Knowledge Base Revision: Preliminary Report.* Actes 7ième AAAI. Philadelphia (PA US). pp.475-479. 1988.

[Ducournau&95] R. Ducournau, M. Habib, M. Huchard, M. Mugnier, A. Napoli. *Le point sur l'héritage multiple*. Techniques et Sciences Informatiques. 14(3):309-345. 1995.

[Ducournau96a] R. Ducournau. *Les incertitudes de la classification incertaine*. Actes 3ième Langages et Modèles à Objets. Leysin (CH). pp.183-200. 1996.

[Ducournau96b] R. Ducournau. *Des langages objets aux logiques terminologiques : les systèmes classificatoires*. Rapport de recherche LIRMM n°96030. 1996.

[Eiter&92] T. Eiter, G. Gottlob. *On the complexity of propositional knowledge base revision, updates, and couterfactuals.* Artificial Intelligence. 57(2-3):227-270. 1992

[Euzenat95] J. Euzenat. *Building consensual knowledge bases : context and architecture.* Towards very large knowledge bases. Actes KB&KS95. Enschede (NL). pp.143-155. 1995.

[Euzenat98] J. Euzenat. *Représentation de connaissance par objets.* Langages et modèles à objets : états et perspectives. R. Ducournau, J. Euzenat, G. Masini, A. Napoli (éds.).1998. A paraître.

[Fagin&83] R. Fagin, J. Ullman, M. Vardi. *On the semantics of updates in databases.* 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. Atlanta (GA US). pp.352-365. 1983.

[Foo95] N. Foo. *Ontology revision*. Actes ICCS-95. Santa Cruz (CA US). LNCS 954. pp.16-31. 1995.

[Friedman&95] N. Friedman, J. Halpern. *Conditional logics of belief change*. Actes 12ième AAAI. Seatlte (WA US). pp.915-921. 1995.

[Friedman&96] N. Friedman, J. Halpern. *Belief revision: A critique*. Actes 5ième International Conference on Principles of Knowledge Representation and Reasoning (KR'96). Cambridge (MA US). pp.421-431. 1996.

[Gärdenfors&88] E. Gärdenfors, P. Makinson. *Revision of knowledge systems using epistemic entrenchment.* Actes 2nd Workshop on theoretical aspects of reasoning about knowledge. Los Altos (CA US). 1988.

[Gärdenfors&93] E. Gärdenfors, P. Makinson. *Nonmonotonic inferences based on expectations*. Artificial Intelligence. 65(2):197-245. 1993.

[Ginsberg86] M. Ginsberg. *Counterfactuals*. Artificial Intelligence. 30(1):35-79. 1986.

[Hansson96] S. Hansson. *A test battery for rational database updating*. Artificial Intelligence. 82:341-352. 1996.

[Johnson90] D. Johnson. *A catalog of complexity class*. Handbook of Theoretical Computer Science. Vol. A. J. van Leeuwen (éds). pp.67-161. MIT Press. 1990.

[Kakas&90] A. Kakas, P. Mancarella. *Database updates through abduction*. Actes 16ième VLDB Conference. Brisbane (Australia). 1990.

[Katsuno&91a] H. Katsuno, A. Mendelzon. *Propositionnal knowledge base revision and minimal change.* Artificial Intelligence. 52:263-294. 1991.

[Katsuno&91b] H. Katsuno, A. Mendelzon. *On the difference between updating a knowledge base and revising it.* Actes 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge (MA US). pp.387-394. 1991.

[Kifer&95] M. Kifer, G. Lausen, J. Wu. *Logical foundations of object-oriented and frame-based languages*. Journal of the ACM. 42(4):741-843. 1995.

[Lévy94] F. Lévy. *A survey of belief revision and updating in classical logic*. Special issue on Revision and Updating in Knowledge Base. Léa Sombé (eds.). International Journal of Intelligent Systems. 9(1):29-59. 1994.

[Lozinskii94] E. Lozinskii. *Resolving contradictions : a plausible semantics for inconsistent systems.* Journal of Automated Reasoning. 12(1):1-31. 1994.

[Mariño93] O. Mariño. R*aisonnement classificatoire dans une représentation à objets multi-points de vue*. Thèse d'informatique. Université Joseph Fourier. Grenoble (FR). 1993.

[Masini&90] G. Masini, A. Napoli, D. Colnet, D. Léonard, K. Tombre. *Les langages à objets*. InterEditions. Paris (FR). 1990.

[Minski75] M. Minski. *A framework for representing knowledge*.The psychology of computer vision. P.H. Winston (éd). McGrawHill. New-York (NY US). pp.156-189. 1975.

[Nebel90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*. LNCS 422. 1990.

[Nebel92] B. Nebel. *Syntax-based Approaches to Belief Revision*. in P. Gärdenfors (ed.). Belief Revision. Cambridge Tracts in Theoretical Computer Science. 29. Cambridge University Press. Cambridge (UK). pp.52-88. 1992.

[Odberg94] E. Odberg. *MultiPerspectives: The classification dimension of schema modification management for object-oriented databases*. Technology of Object-Oriented Languages and Systems (TOOLS USA '94).Santa Barbara (CA US). 1994.

[Papini&94] O. Papini, A. Rauzy. *Révision : mettons un bémol.* Actes 9ième RFIA. Paris. pp.477-485. 1994.

[Papini&95] O. Papini, A. Rauzy. *Révision in Extended Propositional Calculus.* Actes ECSQARU'95. J. Kholas (éds). LNAI 946. pp.328-335. 1995.

[Papini96] O. Papini. *Knowledge revision.* Actes JFPLC'96 (Journées Francophones de Programmation Logique et programmation par Contraintes). Clermont-Ferrand. 1996.

[Penney&87] J. Penney, J. Stein. *Class modification in the gemStone object-oriented DBMS*. Actes 2nd OOPSLA. Orlando (FL US). pp.111-117. 1987.

[Poole88] D. Poole. *A logical framework for default reasoning.* Artificial Intelligence. 36:81-132. 1988.

[Quilian68] M. Quilian. *Semantic memory.* Semantic information processing. MIT Press Cambridge (MA US). pp.227-270. 1968.

[Reiter80] R Reiter. *A logic for default reasoning.* Artificial Intelligence. 13(1):81-132. 1980.

[Satoh88] K. Satoh. *Nonmonotonic reasoning by minimal belief revision.* Actes International Conference on Fifth Generation Computer System. pp.455-462. Tokyo (JP). 1988.

[Sherpa95] Projet Sherpa. *TROPES 1.0 reference manual*. Rapport interne INRIA Rhône-Alpes. Grenoble (FR). 1995.

`ftp://ftp.inrialpes.fr/pub/sherpa/rapports/tropes-manual.ps.gz`

[Sombé94] Léa Sombé. *A glance at revision and updating in knowledge bases*. Special Issue on Revision and Updating in Knowledge Bases. Léa Sombé (eds.). International Journal of Intelligent Systems. 9(1):1-27. 1994.

[Sowa84] J. Sowa. *Conceptual structures. Information processing in mind and machine.* Addison-Wesley Publishing. 1984.

[Theodoratos95a] D. Theodoratos. *Deduction and consistency checking in object-oriented schema structures viewed as logical theories*. Rapport de recherche R042. ERCIM. Rocquencourt (FR). 1995.

[Theodoratos95b] D. Theodoratos. *Updating object-oriented schema structures viewed as logical theories*. Rapport de recherche R043. ERCIM. Rocquencourt (FR). 1995.

[Weydert92] E. Weydert. *Relevance and revision : about generalizing syntax-based belief revision*. LNCS 633. pp.126-138. 1992.

[Winslett88] M. Winslett. *Reasoning about action using a possible model approach.* Actes 7ième AAAI. Philadelphia (PA US). pp.89-93. 1988.

[Wröbel93] S. Wröbel. *On the proper definition of minimality in specialization and theory revision*. LNCS (LNAI) 667. Actes European Conference on Machine Learning. Vienna (AU). pp.65-82. 1993.

[Wröbel94] S. Wröbel. *On the proper definition of minimality in specialization and theory revision*. Machine Learning. Vol. 14. pp.169-191. 1994.