Adaptation d'un processus de construction d'abstractions basé IDM à des modèles bi-niveaux éléments / méta-éléments Application aux logiques de description

Xavier Dolques, Jean-Rémy Falleri, Marianne Huchard, Clémentine Nebut

Adaptation d'un processus de construction d'abstractions basé IDM à des modèles bi-niveaux éléments / méta-éléments Application aux logiques de description

Xavier Dolques*, Jean-Rémy Falleri*, Marianne Huchard*, Clémentine Nebut*

*LIRMM, Univ Montpellier 2 et CNRS
161 rue Ada 34392 MONTPELLIER CEDEX 5
{dolques, falleri, huchard, nebut}@lirmm.fr,

Résumé. L'Analyse Relationnelle de Concepts (ARC) permet la découverte d'abstractions dans différents artéfacts logiciels : code Java ou modèle UML par exemple. Une approche utilisant le paradigme d'Ingénierie Dirigée par les Modèles (IDM) appliqué à l'ARC permet d'obtenir un processus d'abstraction générique : il est possible de construire des abstractions pour n'importe quel artéfact d'entrée (UML ou Java par exemple) en paramétrant simplement l'approche pour le type d'artéfact d'entrée. Cette approche a jusqu'ici été appliquée uniquement à des artéfacts à un niveau (modèles de classes essentiellement). Nous proposons ici une étude pour l'application à des artéfacts à deux niveaux, mélangeant des éléments et des méta-éléments (mélange de classes et d'instances, ou d'associations et de liens par exemple). Cette étude est appliquée aux logiques de description pour lesquelles nous avons développé une extension de l'outil ARC-IDM.

1 Introduction

La présence d'abstractions dans les programmes et les modèles en facilite la compréhension et la maintenance. Parmi les approches de recherche automatique d'abstractions existantes, on trouve l'analyse relationnelle de concepts (ARC), qui est une extension de l'analyse formelle de concepts (AFC) prenant en compte les relations entre les entités analysées, alors que l'AFC se limite aux attributs binaires des entités. Les entités analysées peuvent être par exemple des classes au sens des langages à objets, ou des individus au sens des logiques de description.

Pour appliquer l'ARC, il faut tout d'abord coder les entités à analyser sous forme de tables nommées contextes qui regroupent les informations sur les attributs des entités et sur les relations existant entre elles. Ensuite, l'ARC proprement dite permet de construire des treillis de concepts (aussi appelés treillis de Galois) contenant les abstractions découvertes, abstractions qui sont hiérarchisées suivant un ordre partiel représentant la spécialisation. Il reste ensuite à décoder le treillis de Galois vers le langage des entités de départ.

Une approche basée sur l'ingénierie des modèles a été proposée (Arévalo et al., 2006; Falleri et al., 2007) pour capitaliser le cœur du processus ARC, et fournir un mécanisme générique

de codage en contextes et de décodage de treillis, qu'il suffit de paramétrer pour l'adapter à différents langages (UML, Java, OWL, etc.). Dans ces travaux, pour appliquer un processus ARC, il est donc nécessaire de fournir en plus du modèle à restructurer : le métamodèle d'entrée (correspondant au métamodèle des modèles à restructurer, par exemple le métamodèle UML), ainsi que des informations sur les méta-éléments de ce métamodèle à prendre en compte pour y appliquer l'ARC (par exemple, le nom des classes, le rôle des associations, etc). Ces travaux étaient jusqu'ici appliqués pour restructurer des modèles de classes : découverte de super-entités en UML, Java, et Ecore. La contribution de cet article est l'étude de l'application de l'approche ARC-IDM pour restructurer des modèles bi-niveaux où cohabitent entités et méta-entités. Ce type de modèle se rencontre fréquemment lorsqu'on souhaite présenter un concept et son instance dans le même modèle, comme par exemple dans les diagrammes d'instance d'UML (OMG, 2004), les ressources RDFS (Brickley et Guha, 2004) ou les ontologies conformes à ODM (OMG, 2007) le métamodèle d'OWL. Plus précisément, nous nous intéressons à la manière d'appliquer cette approche ARC-IDM sur des individus au sens des logiques de description. Il s'agit donc, non plus de restructurer un modèle de classes, mais de chercher des abstractions parmi des individus, chaque individu étant typé par une classe. La difficulté est donc que nous devons gérer deux niveaux d'entités : le niveau classiquement appelé M0 où se situent les individus, et le niveau M1 où se situent les classes. La cohabitation dans un modèle (que nous appelerons bi-niveaux) d'individus et de classes rend plus complexe l'application du processus ARC-IDM. Nous montrons dans cet article deux façons d'appliquer ARC-IDM. La première se calque sur l'application mono-niveau, et utilise comme métamodèle d'entrée le métamodèle « standard » des modèles à restructurer, qui inclut donc les deux niveaux *Individu* et *Classe*. Cette solution donne de mauvais résultats que nous analyserons. Nous proposons ensuite une deuxième application d'ARC-IDM, où le métamodèle d'entrée n'est plus le métamodèle des modèles à restructurer mais une hybridation entre le modèle d'entrée et le métamodèle d'entrée. Nous montrons également comment cette hybridation est construite automatiquement et de manière générique.

Cet article se poursuit de la manière suivante : la section 2 fournit un bref aperçu de l'Analyse Relationnelle de Concepts (ARC) et de son application au sein d'une approche IDM (ARC-IDM), ainsi que les éléments des langages de description nécessaires à la compréhension de l'article. La section 3 décrit la manière naïve d'appliquer ARC-IDM, expose les problèmes qu'elle engendre puis la solution plus efficace que nous proposons. La section 4 aborde l'implémentation de cette solution. Nous concluons dans la section 5.

2 Contexte des travaux : Analyse relationnelle de concepts et Logiques de Description

Les travaux présentés dans cet article se basent sur une approche existante d'Analyse Relationnelle de Concepts basée sur l'Ingénierie Dirigée par les Modèles (IDM). Nous résumons dans cette section ces travaux antérieurs et introduisons les concepts des Logiques de Description nécessaires à la compréhension de l'application de cette approche (et de son extension que nous proposons ici) aux Logiques de Description.

2.1 Analyse Relationnelle de Concepts

L'Analyse Formelle de Concepts (AFC, Birkhoff (1940); Barbut et Monjardet (1970); Ganter et Wille (1999)) et l'extension qui nous intéresse ici, en l'occurrence l'Analyse Relationnelle de Concepts (ARC, Huchard et al. (2007a)), appartiennent à une branche de la théorie des treillis qui s'intéresse à la production de regroupements pertinents d'objets sur la base de caractéristiques communes. L'ARC s'attache plus particulièrement à la prise en compte de liens entre les objets dans le processus de regroupement.

Analyse Formelle de Concepts L'AFC prend comme point de départ un « contexte formel » qui inclut une relation binaire reliant les objets à leurs attributs. La figure 1 présente un exemple de contexte formel, décrivant quelques objets de la nature par des caractéristiques.

Définition 1 (Contexte Formel) *Un contexte formel est un triplet $K = (O, A, I)$, où O et A sont respectivement l'ensemble des objets et celui des attributs (caractéristiques), et $I \subseteq O \times A$ est une relation exprimant que $\forall (o, a) \in I$, a est un attribut de l'objet o.*

	Thing	Vegetal	Lieu	Animal
baie		X		
montagne			X	
mouton				X
lichen		X		
loup				X
lapin				X
ours				X
herbe		X		

FIG. 1 – *Le contexte K_{nature} décrit des objets de la nature.*

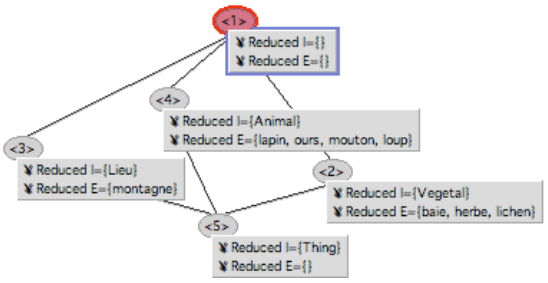

FIG. 2 – *Le treillis L_{nature} regroupe les objets en concepts.*

Un concept associe un ensemble d'objets à l'ensemble d'attributs que ces objets partagent.

Définition 2 (Concept) *Un concept est une paire (X, Y) avec $X \subseteq O$, $Y \subseteq A$ et*
$X = \{o \in O | \forall y \in Y, (o, y) \in I\}$ est l'extension (objets couverts),
$Y = \{a \in A | \forall x \in X, (x, a) \in I\}$ est l'intension (attributs partagés).

Par exemple, pour le contexte K_{nature}, le couple $(\{baie, herbe, lichen\}, \{Vegetal\})$ est un concept, qui apparaît avec le numéro 2 sur le schéma de la figure 2[1].

Les concepts sont ordonnés par spécialisation selon l'ordre partiel suivant : c_1 spécialise c_2 si l'extension de c_1 est incluse dans celle de c_2 (et inversement, l'intension de c_2 est incluse dans celle de c_1). Cet ordre munit l'ensemble des concepts d'une structure de treillis présentée figure 2 avec une simplification : un objet (resp. attribut) apparaissant dans l'étiquette d'un concept est hérité par tous les concepts qui lui sont supérieurs (resp. inférieurs).

[1]Le numéro des concepts ne porte pas de sémantique, il correspond à l'ordre de construction des concepts par l'outil Galicia que nous utilisons ici (Valtchev et al., 2003).

Comme on le voit, l'AFC classique décrit les objets de manière très simple (par des attributs binaires) et l'analyse doit souvent être complétée par des techniques permettant de traiter des situations plus réalistes. L'ARC est l'une de ces extensions, qui prend en compte une description des objets composée de liens entre ces objets.

Analyse Relationnelle de Concepts Parmi les approches permettant de traiter les informations de nature relationnelle, l'ARC, (Huchard et al., 2007a) utilise une représentation naturelle basée sur plusieurs tables, les unes représentant plusieurs catégories d'objets et leur description par des attributs binaires (que nous continuerons à nommer des contextes formels) et les autres représentant des relations entre les objets des différentes catégories (nous les appellerons des contextes relationnels). Le tout forme une famille de contextes relationnels : dans notre exemple, nous nous limiterons à un contexte formel (K_{nature}) et à deux contextes relationnels, R_{mange} et R_{vit}, présentés respectivement par les figures 3 et 4.

	baie	montagne	mouton	lichen	loup	lapin	ours	herbe
baie								
montagne								
mouton								X
lichen								
loup			X			X		
lapin								X
ours	X		X	X				
herbe								

FIG. 3 – *Le contexte relationnel R_{mange}.*

	baie	montagne	mouton	lichen	loup	lapin	ours	herbe
baie								
montagne								
mouton		X						
lichen								
loup		X						
lapin		X						
ours		X						
herbe								

FIG. 4 – *Le contexte relationnel R_{vit}.*

Définition 3 (Famille de Contextes Relationnels (RCF)) *Une famille de contextes relationnels $\mathcal{R}$ est un couple (K, R). K est un ensemble de contextes formels $K_i = (O_i, A_i, I_i)$, R est un ensemble de contextes relationnels $R_j = (O_k, O_l, I_j)$ (O_k et O_l sont les ensembles d'objets des contextes K_k et K_l de K).*

De nouvelles abstractions émergent par itération d'une étape de construction des treillis de concepts suivie d'une étape de concaténation des contextes formels avec les contextes relationnels correspondant enrichis par les concepts créés à l'étape précédente.

Etape d'initialisation. Les treillis sont construits à cette étape en utilisant l'AFC classique : pour chaque contexte formel K_i, un treillis $\mathcal{L}_i^0$ est créé (il s'agit du treillis de la figure 2 dans notre cas).

Etape n+1. Pour chaque contexte relationnel $R_j = (O_k, O_l, I_j)$, un contexte enrichi $R_j^s = (O_k, A, I)$ est créé. A correspond aux concepts du treillis $\mathcal{L}_l^n$. Prenant comme exemple R_{mange}, nous obtenons $R_{mange}^s = (O_{nature}, L_{nature}, I)$.

La relation d'incidence I contient l'élément (o, a) si $S(R(o), Extension(a))$ est $vrai$. La fonction S est un opérateur de *scaling*. Les deux opérateurs de scaling qui nous intéresseront ici sont $S_\exists(R(o), Extension(a))$, qui est $vrai$ si et seulement si $\exists x \in R(o), x \in Extension(a)$,

Knature

	Thing	Vegetal	Lieu	Animal
baie		X		
montagne			X	
mouton				X
lichen		X		
loup				X
lapin				X
ours				X
herbe		X		

R(s)mange

	C1	C2	C3	C4	C5
baie					
montagne					
mouton	X	X			
lichen					
loup	X			X	
lapin	X	X			
ours	X				
herbe					

R(s)vit

	C1	C2	C3	C4	C5
baie					
montagne					
mouton	X		X		
lichen					
loup	X		X		
lapin	X		X		
ours	X		X		
herbe					

FIG. 5 – *Le contexte K_{nature} étendu par les deux contextes relationnels R^s_{mange} et R^s_{vit}.*

et $S_{\forall\exists}(R(o), Extension(a))$, qui est $vrai$ ssi $\forall x \in R(o), x \in Extension(a) \wedge \exists x \in R(o), x \in Extension(a)$.

Examinons tout d'abord le cas de l'opérateur $S_{\exists}$ appliqué à R_{mange}. L'étape d'initialisation a permis de découvrir l'abstraction représentée par le concept C_2 (les végétaux). Comme nous avons $(baie) \in R_{mange}(ours)$ avec $baie \in Extension(C_2)$, $(ours, C_2) \in I$. De même, nous aurons $(lapin, C_2) \in I$. On traduit ainsi le fait que l'ours et le lapin mangent au moins une sorte de végétal.

Puis étudions l'opérateur $S_{\forall\exists}$. Comme nous avons $(mouton) \in R_{mange}(ours)$ avec $mouton \notin Extension(C_2)$, cette fois $(ours, C_2) \notin I$. Par contre, puisque $R_{mange}(lapin) = \{herbe\} \subseteq Extension(C_2)$, nous avons toujours $(lapin, C_2) \in I$. On traduit ainsi le fait que le lapin ne mange que des végétaux, alors que l'ours ne mange pas que des végétaux puisqu'il mange aussi les moutons.

L'application de l'AFC sur $K_k \cup \{R^s_j = (O_k, A, I)\}$ crée de nouveaux concepts qui sont ajoutés à $\mathcal{L}^n_k$ pour obtenir $\mathcal{L}^{n+1}_k$. Par exemple, en continuant de travailler avec l'opérateur $S_{\forall\exists}$, on voit se former sur le contexte K_{nature} étendu par les deux contextes relationnels R^s_{mange} et R^s_{vit} (figure 5) le concept $(\{mouton, lapin\}, \{Animal, mange : C1, C2, vit : C1, C3\})$ qui représente les objets qui sont des animaux, ne mangent que des végétaux et ne vivent que dans des lieux (ici de montagne car on n'a pas développé l'exemple plus avant). Il faut noter que ce concept ne pouvait pas apparaître lors de la précédente construction de concepts (étape d'initialisation). Le processus s'arrête quand les treillis d'une étape n sont équivalents (aux étiquettes près) à ceux de l'étape $n-1$: en effet il n'y aura plus d'enrichissement possible.

2.2 Analyse relationnelle de concepts dans un cadre IDM

Nous nous intéressons ici à une conception du processus ARC basée sur l'ingénierie dirigée par les modèles. Nous résumons cette approche, développée dans (Arévalo et al., 2006; Falleri et al., 2007). Précédemment divers prototypes (Valtchev et al., 2003; Dao et al., 2006; Seuring, 2005) avaient été proposés pour mettre en œuvre une démarche ARC et testés notamment avec différentes versions des diagrammes de classes UML. Or les applications de l'ARC sont multiples et variées (modèles de composants, logiques de description, etc.), et chacune entraîne la nécessité de convertir les données du domaine vers les familles de contextes relationnels, de

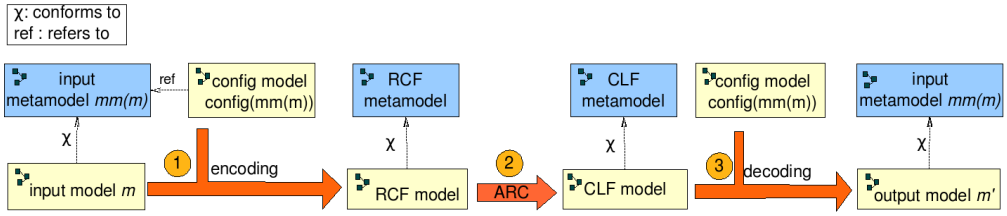

FIG. 6 – *Approche ARC-IDM.*

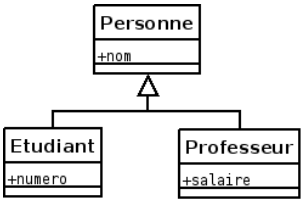

FIG. 7 – *Un modèle UML.*

paramétrer diversement la démarche ARC et de revenir des familles de treillis vers des données métier.

La figure 6 donne un aperçu de l'approche proposée pour aborder ce problème. Les entrées d'ARC-IDM sont, outre le modèle *m* à restructurer, le métamodèle de *m*, *mm(m)* et un modèle *config(mm(m))* qui définit quels sont les méta-éléments de *mm(m)* à prendre en compte pour la restructuration. La conversion du modèle en une famille de contextes relationnels se fait de la manière suivante :

- Un contexte formel est créé pour chaque méta-classe indiquée dans config(mm(m)). Les attributs binaires utilisés dans le contexte sont choisis parmi les attributs de la méta-classe. Finalement, il est possible de spécifier un lien de spécialisation-généralisation pour une méta-classe donnée. Ce lien permet de calculer les attributs relationnels hérités par les éléments. Par exemple, pour UML, il peut y avoir un contexte formel pour les classes, un pour les propriétés. L'attribut nom de la méta-classe Property est utilisé pour décrire les propriétés. La relation generalization.general permet de récupérer les attributs introduits par les super-classes d'une classe.
- Un contexte relationnel est créé pour chaque méta-relation indiquée dans config(mm(m)). Par exemple, pour UML, il peut y avoir un contexte relationnel entre les classes et les propriétés qui représente la relation ownedAttribute.

Pour clarifier le fonctionnement de cette conversion, en voici un exemple sur le modèle UML de la figure 7. Le modèle de configuration utilisé est le suivant :

```
FormalContexts:
    * Class: attribute = [], specializationLink = generalization.general
    * Property: attributes = [name], specializationLink = redefinedProperty
RelationalContexts:
    * ownedAttribute: source = Class, target = Property
```

La famille de contextes générée à l'aide de cette configuration est présentée à la figure 8. Le retour des familles de treillis vers le modèle métier se fait de manière symétrique.

cls_personne
cls_professeur
cls_etudiant

	name : nom	name : salaire	name : numero
prop_nom	X		
prop_numero			X
prop_salaire		X	

	prop_nom	prop_numero	prop_salaire
cls_personne	X		
cls_professeur	X		X
cls_etudiant	X	X	

FIG. 8 – *Famille de contextes générée à partir de l'exemple de la figure 7. On y trouve le contexte formel* Class *(en haut à gauche), le contexte formel* Property *(en haut à droite) et le contexte relationnel* OwnedProperty *(en bas) qui a pour source* Class *et pour cible* Property.

2.3 Logiques de description

Les logiques de description (Baader et Nutt, 2003) permettent la représentation des connaissances à l'aide de *Concepts* qui sont les différents types d'éléments que l'on peut trouver, d'*Individus*, qui sont les éléments du domaine, et de *Rôles* qui sont les relations que l'on peut définir entre les individus. Comme montré dans Huchard et al. (2007b), il est approprié de leur appliquer l'ARC de manière à faire émerger de nouveaux concepts.

La définition d'une hiérarchie de concepts en logique de description forme ce que l'on appelle la boîte terminologique (TBox). Les concepts sont des concepts atomiques (comme $Vegetal$, $Animal$), le concept universel ($\top$), le concept vide ($\bot$) ou définis par une expression utilisant différents constructeurs en fonction de la logique choisie, tels que la négation ou la conjonction (notée $\sqcap$) de concepts. Nous serons ici plus spécialement intéressés par les constructeurs qui font intervenir des rôles : la quantification universelle de rôle ($\forall R.C$, avec R un rôle et C un concept) et la quantification existentielle de rôle ($\exists R.C$). Le langage de description se rapprochant alors le plus de nos besoins est $\mathcal{FL}^{-}\mathcal{E}$ puisqu'il est défini par ces constructeurs. Si $mange$ est un rôle, on peut définir le concept $Herbivore$ grâce à l'expression $Herbivore := Animal \sqcap \exists mange.\top \sqcap \forall mange.Vegetal$. On traduit par cette définition le fait qu'un herbivore est un animal qui mange au moins une chose et qui ne mange que des végétaux. On retrouve dans la boîte des assertions – ou ABox – les instanciations des concepts définis dans la boîte terminologique. Un individu (par exemple $herbe$) est défini par son type (par exemple $Vegetal(herbe)$), qui est un concept de la TBox, et un rôle est défini par un ensemble de couples d'individus comme $mange(lapin, herbe)$.

3 Adaptation d'ARC-IDM pour la restructuration de modèles bi-niveaux

Ainsi qu'expliqué à la section précédente, l'approche ARC-IDM propose un mécanisme d'encodage générique en famille de contextes. Pour appliquer l'ARC à un modèle, il suffit donc d'en donner le métamodèle et la configuration ARC pour ce métamodèle. ARC-IDM a jusqu'ici été appliqué sur des modèles contenant des entités d'un seul niveau, c'est-à-dire ne mélangeant pas entités et méta-entités. Nous montrons ici comment appliquer ARC-IDM sur des modèles bi-niveaux, mélangeant des entités et des instances d'entités. Ce type de modèle est relativement courant, on peut en retrouver dans les modèles cherchant à représenter des instances comme ODM (OMG, 2007) qui permet de modéliser des ontologies ou dans les diagrammes d'instances d'UML (OMG, 2004). Nous présentons cette application avec l'exemple des logiques de description : nous cherchons donc à restructurer des modèles contenant des individus basés sur des classes. Ces modèles se composent ainsi d'une TBox et d'une ABox, nous les appellerons par la suite des modèles TAB. Nous montrons dans cette section pourquoi une adaptation naïve calquée sur celle des modèles mono-niveau n'est pas appropriée, et quelle adaptation doit donc être mise en œuvre. Cette section est illustrée par l'exemple des animaux utilisé à la section précédente.

3.1 Adaptation naïve, calquée sur la modélisation mono-niveau

Une première adaptation qui a été envisagée consiste à calquer la restructuration des modèles bi-niveaux sur celle des modèles mono-niveau. Ainsi, pour restructurer un modèle TAB contenant des individus typés par des classes et liés par des instances de relations typées par des relations, nous fournissons à ARC-IDM un métamodèle faisant cohabiter les classes, les individus, les relations et les instances de relation, ainsi qu'illustré en haut de la figure 9. Notons que nous travaillons avec un métamodèle simplifié des logiques de description. Le modèle d'exemple des animaux sur lequel nous allons travailler est donc une instance de ce métamodèle, un extrait en est proposé dans le bas de la figure 9, sur lequel on ne voit apparaître que les animaux, qui habitent tous dans la montagne.

Il nous faut fournir également un modèle de configuration pour ce métamodèle (nous rappelons que cette configuration consiste à sélectionner les éléments du métamodèle qui seront utilisés pour l'ARC). Toutes les entités de notre métamodèle correspondent à un contexte formel, tandis les liens inter-entités sont à l'origine des contextes relationnels. On choisit ensuite pour chaque contexte relationnel un opérateur de scaling.

Le modèle de configuration précise qu'on prend en compte les 4 types d'entités : *Classe*, *Individu*, *Instance de relation* et *Relation*. Les quatre contextes formels correspondants seront donc générés. On précise également dans le modèle de configuration que l'on prend en compte :

- l'association *type*[2] liant *Individu* à *Classe*. On lui associe l'opérateur de scaling $S_\exists$. Le contexte relationnel R_{type} sera donc créé, associé à l'opérateur $S_\exists$;
- l'association *est source de* liant *Individu* à *Instance Relation* (cela générera le contexte $R_{estsourcede}$). On lui associe l'opérateur $S_{\forall\exists}$;
- l'association *a pour cible* liant *Instance Relation* à *Individu*. Le contexte relationnel $R_{apourcible}$ sera donc créé, associé à l'opérateur $S_\exists$;

[2] *type* est en fait un rôle de cette association.

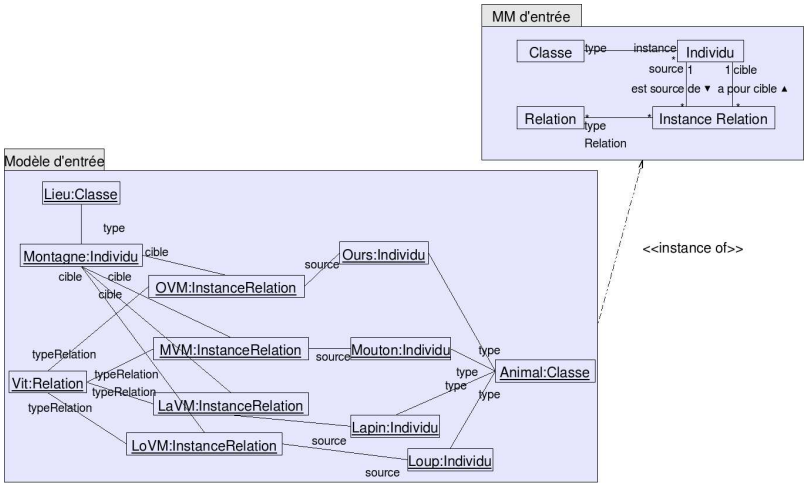

FIG. 9 – *Adaptation naïve pour l'exemple des animaux.*

– l'association *type relation* liant *Instance Relation* à *Relation*. Le contexte relationnel $R_{typerelation}$ sera donc créé, associé à l'opérateur $S_\exists$.

Ainsi, en utilisant le modèle de configuration, on a un contexte formel unique pour toutes les *instances de relation*, qui sont reliées aux *relations* par une relation inter-contextes, représentée par le contexte relationnel *type de relation* que l'on peut voir dans la figure 10. Or s'il est possible d'obtenir certaines abstractions par cette modélisation, d'autres qui pourtant peuvent émerger lorsqu'on applique l'ARC de manière classique, c'est-à-dire en remplissant les différents contextes sans tenir compte de la modélisation des données d'entrée, ne peuvent pas émerger avec cette configuration.

Les problèmes inhérents à cette modélisation surviennent lorsqu'on utilise un opérateur de scaling différent de $S_\exists$. Référons-nous à l'exemple : nous nous intéressons à des animaux, que nous relions à leur habitat et régime alimentaire. En utilisant l'ARC nous espérons voir apparaître certaines abstractions qui paraissent évidentes, notamment un découpage entre les animaux herbivores, carnivores et omnivores, découpage ébauché à la section 2.1. La définition que l'on peut donner pour ces concepts est : un herbivore est un animal tel que quoi qu'il mange, c'est de type végétal ; un carnivore est un animal tel que quoi qu'il mange, c'est de type animal ; enfin un omnivore n'appartient à aucune de ces catégories puisque son régime alimentaire ne se limite à aucun des deux régimes carnivore ou herbivore.

On voit que ces définitions nécessitent l'opérateur $S_{\forall\exists}$, plutôt que $S_\exists$, car avec l'opérateur $S_\exists$, on aurait par exemple pu définir l'ensemble des animaux qui mangent des végétaux, mais pas l'ensemble des animaux qui ne mangent que des végétaux. Malheureusement, l'utilisation de cet opérateur ne donne pas les résultats escomptés. En effet, nous avons toutes les *instances de relations* qui sont représentées par une même métaclasse dans le métamodèle. Cela implique que le contexte relationnel aura pour cible un contexte formel contenant l'ensemble des *instances de relation*. Or l'opérateur de scaling s'applique sur tous les éléments du contexte relationnel. La figure 11 montre une illustration du problème qui se pose.

	ir0	ir1	ir2	ir3	ir4	ir5	ir6	ir7	ir8	ir9	ir10
baie											
montagne											
mouton	X	X									
lichen											
loup			X	X	X						
lapin						X	X				
ours								X	X	X	X
herbe											

	baie	montagne	mouton	lichen	loup	lapin	ours	herbe
ir0								X
ir1		X						
ir2						X		
ir3			X					
ir4		X						
ir5								X
ir6		X						
ir7				X				
ir8	X							
ir9			X					
ir10		X						

	mange	vit
ir0	X	
ir1		X
ir2	X	
ir3	X	
ir4		X
ir5	X	
ir6		X
ir7	X	
ir8	X	
ir9	X	
ir10		X

FIG. 10 – *Contexte relationnels des relations* est source de *(en haut à gauche),* a pour cible *(en bas à gauche) et* type de relation *(à droite).*

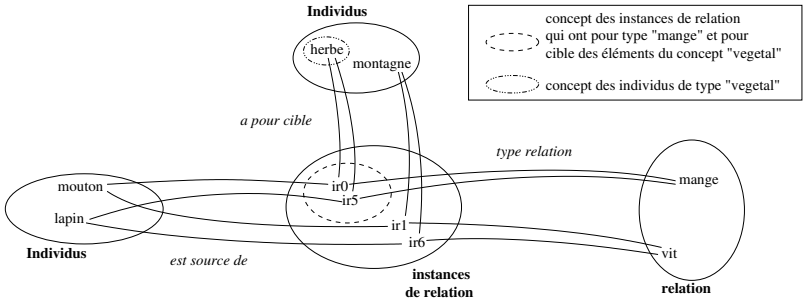

FIG. 11 – *Exemple illustrant le problème posé par une approche classique sur les modèles bi-niveaux. Le contexte des* Individus *est présent deux fois dans le schéma pour des raisons de facilité de lecture, mais il s'agit bien du même contexte où seuls les individus qui nous intéressent sont représentés.*

L'ARC est un processus qui se déroule de manière itérative, pour pouvoir découvrir des concepts dépendant de ceux déjà obtenus. Dans notre exemple, après la première itération des concepts se sont formés dans le contexte des *individus*. L'un d'eux est le concept des *individus* de type *végétal*, qui contient l'*individu herbe* et que l'on nommera par la suite le concept des *végétaux*. À l'itération suivante de nouveaux concepts sont créés dans le contexte des *instances de relation*. On y trouve le concept des *instances de relation* qui ont *mange* pour *type de relation* et qui *ont pour cible* un élément du concept des *végétaux* découvert dans l'itération précédente. Enfin, à l'itération qui suit le processus va permettre de créer de nouveaux concepts, définis à partir des concepts du contexte *instance de relation* nouvellement créés, dans le contexte des individus grâce à la relation *est source de*. Étudions maintenant l'opérateur $S_{\forall\exists}$ pour la relation *est source de*. Un ensemble d'individus ne pourra définir l'extension d'un nouveau concept que si toutes les *instances de relation* dont ses éléments *sont source* appartiennent à un même concept. On voit dans la figure 11 que l'*individu mouton* n'appartient pas à

un nouveau concept, alors qu'il a la particularité de ne manger que de l'herbe. Ce qui empêche la création de ce nouveau concept, c'est le fait que *mouton est source de ir0* et *ir1*, or *ir1* est de type *vit* et n'appartient pas au même concept que *ir0*.

La source du problème vient de plusieurs facteurs ; le premier est que notre approche, dans un souci de généricité, génère les contextes formels à partir des éléments d'un méta-modèle ; l'autre facteur est l'utilisation de modèle contenant plusieurs niveaux d'abstraction et la relation d'instanciation permettant de transférer d'un niveau à l'autre est définie dans le métamodèle par une simple association. Dans notre cas il est clair qu'il existe une relation d'instanciation entre les *relations* et les *instances de relation*, mais les *relations* ne font pas partie du méta-modèle, donc il n'est pas possible de créer un contexte formel pour chaque *relation*.

Devant les limites de cette adaptation naïve, nous proposons à la section suivante une adaptation plus complexe, mais plus satisfaisante.

3.2 Adaptation pour les modèles bi-niveaux : hybridation du métamodèle d'entrée avec le modèle d'entrée

Comme nous l'avons vu dans la partie précédente, notre problème vient de l'impossibilité d'appliquer l'opération de scaling sur un sous-ensemble seulement des éléments du contexte formel cible d'un contexte relationnel. Dans le cas précis de notre modèle de logique de description, le contexte relationnel en question est *est source de* reliant des *individus* à des *instances de relation*, et les *instances de relations* appartenant à l'ensemble sur lequel on veut appliquer l'opération de scaling ont en commun l'association *type de relation* qui pointe vers la même *relation*. Nous proposons donc de promouvoir une partie du modèle, c'est-à-dire de passer les *relations* dans le métamodèle, sous forme de relation du modèle et de transformer la relation *type de relation* en une relation d'instanciation. Nous hybridons une partie du méta-modèle d'entrée avec une partie du modèle d'entrée, ainsi qu'illustré à la figure 12. Le modèle d'entrée est donc lui aussi modifié, ainsi que le montre le bas de la figure 13. La transformation d'hybridation a pour but de supprimer la réification des relations d'un modèle par le concept *Instance de Relation* car elle n'apparaît pas pertinente pour la recherche de nouvelles abstractions.

Le modèle de configuration suit de la même manière la transformation, on ne prend plus en compte que deux entités du méta-modèle, *Classe* et *Individu*, et les relations prises en compte sont :

- l'association *type* liant *Individu* à *Classe*. On lui associe l'opérateur de scaling $S_{\exists}$. Le contexte relationnel R_{type} sera donc créé, associé à l'opérateur $S_{\exists}$;
- l'association *vit* reliant *Individu* à lui-même. On lui associe l'opérateur de scaling $S_{\forall\exists}$.
- l'association *mange* reliant *Individu* à lui-même. On lui associe l'opérateur de scaling $S_{\forall\exists}$.

Cette solution n'implique pas de modifier le processus de l'ARC proprement dit, nous ne modifions que ses modèles d'entrée. Ces modifications ont pour conséquence que l'on peut obtenir un contexte relationnel pour chaque *relation* du modèle d'origine, soit exactement les contextes des figures 3 et 4. La lecture des tables se trouve facilitée, car les instances de relations sont représentées par une table pour chaque relation, et que cette table contient la source et la cible de l'instance de relation. L'hybridation du modèle a alors pour conséquence de ré-

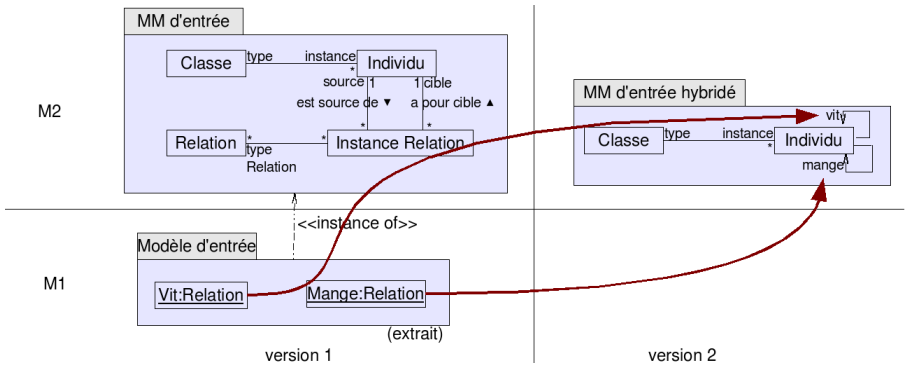

FIG. 12 – *Hybridation du métamodèle.*

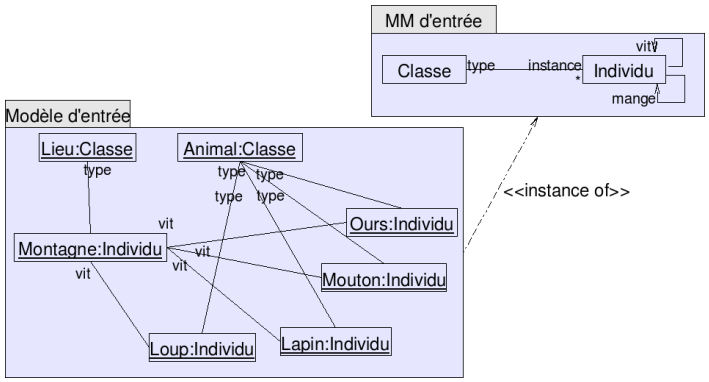

FIG. 13 – *Adaptation hybridée pour l'exemple des animaux.*

tablir le nivellement des éléments du modèle tout en conservant la totalité des informations du modèle d'origine et les relations du modèle d'origine qui étaient sémantiquement des relations d'instance deviennent de vraies relations d'instance. Le modèle hybride accompagné de son méta-modèle est alors un modèle mono-niveau auquel on peut appliquer le processus d'ARC-IDM de manière classique. On voit dans la figure 14 que le problème d'application de l'opérateur de scaling $S_{\forall\exists}$ ne se pose plus, toutes les *instances de relation* qui possédaient le même *type de relation* dans le modèle d'origine sont transformées en relations du méta-modèle de même type. Les nouvelles définitions de concepts dans le contexte des individus pourront ne dépendre que d'une relation particulière. Dans l'exemple on pourra définir un concept auquel appartient *mouton* et qui regroupe tous les individus qui "ne mangent que des végétaux", indépendamment des autres types de relation telles que la relation *vit*. Cette modélisation du problème correspond à ce qu'on peut obtenir avec une application classique de l'ARC (non dirigée par les modèles).

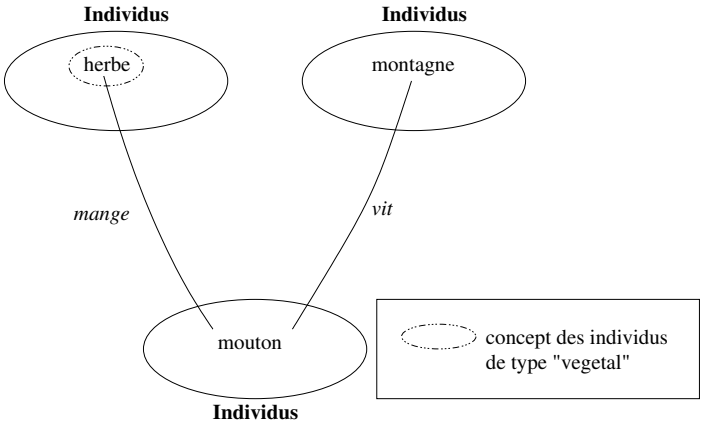

FIG. 14 – *Exemple illustrant comment sont traités les modèles bi-niveaux avec notre nouvelle approche.*

formulation logique	exemple	modélisation naïve	modélisation avancée	ARC originale
$\exists ir \in IR$ et $\forall ir \in IR, ir.typeDeRelation(T)$	concept des animaux qui ne sont reliés à d'autres concepts (au moins un) que par la relation mange	X		
$\exists ir \in IR$ et $\forall ir \in IR, ir.aPourCible(x)$ où $x \in C$	concept des animaux qui ne sont reliés qu'à un ou plusieurs végétaux (il vit dans l'herbe, mange de l'herbe ...)	X		
$\exists ir \in IR$ et $\forall ir \in IR, ir.typeDeRelation(T)$ et $ir.aPourCible(x)$ où $x \in C$	concept des animaux qui ne mangent que des végétaux (au moins un) et qui ne sont pas reliés par d'autres relations à d'autres concepts	X		
$\exists ir \in IR$ de type T donné et $\forall ir \in IR$ de type T donné, $ir.aPourCible(x)$ où $x \in C$	concept des animaux qui ne mangent que des végétaux et qui en mange au moins un		X	X
$\exists ir \in IR$ tel que $ir.typeDeRelation(T)$	concept des animaux qui mangent au moins une chose	X		
$\exists ir \in IR$ tel que $ir.aPourCible(x)$ où $x \in C$	concept des animaux qui sont reliés au moins une fois à des éléments du concept des végétaux(quelque soit la relation)	X		
$\exists ir \in IR$ tel que $ir.typeDeRelation(T)$ et $ir.aPourCible(x)$ où $x \in C$	concept des animaux qui mangent au moins un élément du concept des végétaux	X	X	X
$\exists ir \in IR$ de type T donné tel que $ir.aPourCible(x)$ où $x \in C$	concept des animaux qui mangent au moins un élément du concept des végétaux	X	X	X

TAB. 1 – *Tableau récapitulatif des concepts exprimables avec chacune des techniques.*

3.3 Bilan

L'approche ARC-IDM a pour but d'implémenter de manière la plus proche possible l'ARC originale qui a été présentée dans la section 2.1. Lorsque nous avons étudié les modèles bi-niveaux, nous nous somme aperçus que nous ne pouvions pas obtenir la même expressivité que l'ARC originale dans la définition des abstractions créées avec une adaptation naïve du modèle. Nous montrons dans le tableau 1 que les définitions de concepts, surtout si on utilise l'opérateur $S_{\forall\exists}$, sont différentes selon la manière dont on adapte le modèle à l'ARC-IDM. L'adaptation par hybridation que nous proposons dans cet article est celle qui s'approche le plus de l'ARC originale.

Le tableau 1 liste les différents concepts exprimables dans le contexte des *Individus*. La première colonne montre la formulation logique des différents concepts possibles. Dans chaque cas IR représente l'ensemble des *instances de relation* dont les éléments du concept définis sont sources (l'ensemble des liens sortants du concept que l'on définit).

4 Mise en oeuvre dans la plateforme ARC-IDM

La plate-forme implémentant ARC-IDM utilise le framework de modélisation EMF (Budinsky et al., 2003). Elle se base sur le langage de méta-modélisation Ecore pour lire les modèles et leur métamodèle.

Pour représenter notre modèle de logique de description, nous avons décidé d'utiliser le langage OWL (Web Ontology Language, McGuinness et van Harmelen (2004)). En effet, OWL est très utilisé dans les technologies Web comme langage de représentation de connaissance, et on dispose ainsi de nombreux outils de modélisation pour OWL, ainsi que d'exemples de modèles OWL. De plus, notre métamodèle est inclus, au nommage des entités près, dans celui proposé dans le plugin Eclipse EODM (EODM) qui permet de manipuler les modèles OWL avec EMF. Ainsi, un modèle TAB peut se traduire facilement par simple renommage en un modèle OWL, avec une correspondance entre les Classes et les OWLClass, les Individus et les Individual, etc.

Pour adapter la plateforme ARC-IDM aux modèles TAB, nous devons définir la transformation d'hybridation du métamodèle OWL par un modèle TAB particulier, et, comme pour restructurer n'importe quel type de modèle, définir une configuration précisant les méta-éléments pris en compte pour l'ARC. La figure 15 nous permet de voir à quel niveau de l'ARC-IDM s'applique la transformation d'hybridation, celle-ci se limitant pour le moment à prendre des modèles OWL en entrée.

La transformation d'hybridation est automatique, elle est écrite en Java et utilise EMF pour manipuler les modèles. Elle prend en entrée le métamodèle OWL, un modèle TAB quelconque et le modèle de configuration puis génère à la fois le métamodèle hybridé par les relations du modèle, le nouveau modèle TAB conforme au métamodèle hybridé et la nouvelle configuration.

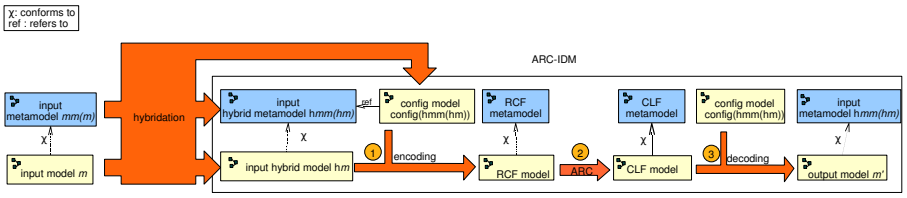

FIG. 15 – *Illustration de l'hybridation dans le processus ARC-IDM.*

Ainsi que montré à la figure 12, la transformation d'hybridation répertorie l'ensemble des relations et les monte d'un niveau d'abstraction pour les faire apparaître dans le métamodèle hybridé. Cette transformation ne prend en compte que les *relations* et les *instances de relation*, laissant intacts les classes et les individus. La transformation transforme également le modèle de configuration. En effet, un modèle de configuration se réfère à un métamodèle donné. Le métamodèle ayant été hybridé, il faut modifier la configuration pour prendre en compte l'hybridation. La configuration obtenue peut ensuite être modifiée selon les besoins de l'utilisateur, notamment pour choisir l'opérateur de scaling d'un contexte relationnel.

Nous avons testé cette transformation sur plusieurs exemples dont les modèles OWL sont disponibles sur http://www.lirmm.fr/~nebut/Publications/ArticleSupplements/LMO2008/ARC_IDM_LD/, et

pour chacun de ces exemples nous avons testé plusieurs types de modèles de configurations, dont celui généré automatiquement. Nous obtenons alors par application de l'ARC des treillis à partir desquels on peut déduire de nouvelles définitions de concepts. Pour l'instant une sortie sous forme de texte est proposée.

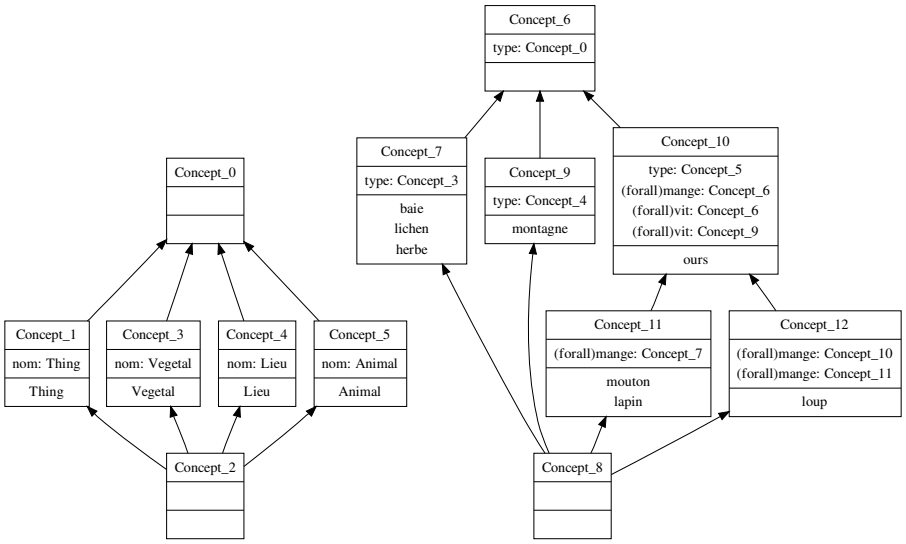

FIG. 16 – *Treillis obtenus en appliquant l'ARC-IDM sur notre exemple. Le treillis de gauche est le treillis des classes, celui de droite est celui des individus. Chaque concept des treillis est partagé en trois parties : la partie du haut représente le nom du concept, généré automatiquement lors du processus ; la partie du milieu représente l'intention du concept définissant ses caractéristiques ; la partie du bas représente l'extension du concept.*

La figure 16 nous montre les treillis obtenus par l'application de l'ARC-IDM sur notre exemple. Pour rendre le treillis plus lisible nous le représentons sous forme de diagramme de Hasse. Nous considérons que tout élément apparaissant dans l'intension d'un concept appartient aussi à l'intension des concepts qui spécialisent ce concept et tout élément apparaissant dans l'extension d'un concept appartient aussi à l'extension des concepts qui généralisent ce concept. Le treillis qui nous intéresse est celui obtenu à partir du contexte des *Individus*, on voit que des concepts ont été créés pour regrouper les éléments de même type : le *Concept_7* regroupe les éléments de type *végétal*, le *Concept_9* ceux de type *lieu* et le *Concept_10* ceux de type *animal*. Le *Concept_11* regroupe les animaux qui ne mangent que des végétaux. Le *Concept_12* regroupe les animaux qui ne mangent que des animaux et plus particulièrement des animaux du *Concept_11*. A partir de l'intension des nouveaux concepts, on peut en déduire une formule logique les définissant de manière automatisée.

Il résulte donc de nos travaux une plateforme pour restructurer des modèles bi-niveaux de manière automatique : de même que pour restructurer les modèles mono-niveau, pour restructurer un modèle bi-niveaux il faut juste configurer l'ARC en fournissant un modèle de configuration. Notre expérience nous montre que la configuration pour les modèles bi-niveaux

est légèrement plus complexe à concevoir que pour les modèles mono-niveau car elle mélange fréquemment différents opérateurs de scaling.

5 Conclusion

La construction d'abstractions par l'Analyse Relationnelle de Concepts (ARC) selon le paradigme de l'Ingénierie Dirigée par les Modèles a été appliquée jusqu'ici à des modèles à un seul niveau d'abstraction, au sens où les modèles ne mélangent pas entités et méta-entités. Dans cet article, nous avons étudié l'utilisation de cette approche pour des modèles bi-niveaux, où cohabitent des méta-entités et leurs instances, en nous appuyant sur l'exemple des logiques de description : nous nous sommes intéressés à des modèles mélangeant des individus et les liens qui les unissent avec les classes des individus et les relations définissant les liens. Nous avons montré qu'une approche naïve calquée sur celle utilisée pour des modèles mono-niveau n'était pas adaptée, et nous avons proposé et implémenté une solution basée sur la promotion d'une partie des modèles bi-niveaux vers leur métamodèle. Cette deuxième solution donne des résultats identiques à ceux obtenus lors d'une application manuelle logique de l'ARC, ce qui était l'objectif visé.

Nous avons travaillé sur une logique de description d'expressivité minimale, notamment nous ne prenons pas en compte la relation de spécialisation pouvant s'appliquer entre les classes, ou les contraintes que l'on peut appliquer sur les sources et les cibles des relations telles qu'on peut les trouver dans OWL. Nous étendrons donc dans de futurs travaux notre approche à des logiques de description plus complexes et travaillerons sur l'intégration des concepts obtenus dans le modèle d'origine. Nous validerons également notre approche en rendant la transformation paramètrable pour l'appliquer sur d'autres modèles bi-niveaux, comme par exemple des modèles UML faisant cohabiter classes et instances, ainsi qu'associations et liens.

Remerciements. Nous remercions Jean-François Baget pour son éclairage sur le domaine des logiques de description ainsi qu'Amedeo Napoli, Mohamed Rouane Hacène et Petko Valtchev qui ont été la source d'inspiration de ce travail.

Références

Arévalo, G., J.-R. Falleri, M. Huchard, et C. Nebut (2006). Building abstractions in class models : Formal concept analysis in a model-driven approach. In *Proc. of the MoDELS'06 conference*, pp. 513–527.

Baader, F. et W. Nutt (2003). Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, et P. F. Patel-Schneider (Eds.), *The Description Logic Handbook : Theory, Implementation, and Applications*, pp. 43–95. Cambridge University Press.

Barbut, M. et B. Monjardet (1970). *Ordre et Classification : Algèbre et Combinatoire*, Volume 2. Hachette.

Birkhoff, G. (1940). *Lattice theory*. American Mathematical Society, Providence, RI, 1st édition.

Brickley, D. et R. V. Guha (2004). RDF Vocabulary Description Language 1.0 : RDF Schema. Technical report, W3C.

Budinsky, F., T. Grose, D. Steinberg, R. Ellersick, E. Merks, et S. Brodsky (2003). *Eclipse Modeling Framework : a developer's guide*. Addison-Wesley Professional.

Dao, M., M. Huchard, M. R. Hacene, C. Roume, et P. Valtchev (2006). Towards Practical Tools for Mining Abstractions in UML Models. In Y. Manolopoulos, J. Filipe, P. Constantopoulos, et J. Cordeiro (Eds.), *ICEIS (3)*, pp. 276–283.

EODM. site du projet EODM, http ://www.eclipse.org/modeling/mdt/ ?project=eodm.

Falleri, J.-R., M. Huchard, C. Nebut, et G. Arévalo (2007). Use of Model Driven Engineering in Building Generic FCA/RCA Tools. In J. Diatta, P. Eklund, et M. Liquière (Eds.), *Proc. of CLA'07 : Fifth International Conference on Concept Lattices and Their Applications*, pp. 225–236.

Ganter, B. et R. Wille (1999). *Formal Concept Analysis, Mathematical Foundations*. Springer, Berlin.

Huchard, M., M. R. Hacene, C. Roume, et P. Valtchev (2007a). Relational concept discovery in structured datasets. *Ann. Math. Artif. Intell. 49*(1-4), 39–76.

Huchard, M., A. Napoli, M. R. Hacène, et P. Valtchev (2007b). Mining description logics concepts with relational concept analysis. In P. Brito, P. Bertrand, G. Cucumel, et F. D. Carvalho (Eds.), *Selected Contributions in Data Analysis and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 259–270. Springer, Berlin.

McGuinness, D. L. et F. van Harmelen (2004). OWL Web Ontology Language Overview. Technical report, W3C.

OMG (2004). UML 2.0 superstructure. Technical report, Object Management Group.

OMG (2007). Ontology definition metamodel. Technical report, Object Management Group.

Seuring, P. (2005). Design and implementation of a UML model refactoring tool. Master's thesis, Hasso-Plattner-Institute for Software Systems Engineering at the University of Postdam. http ://www.lirmm.fr/~huchard/Documents/Papiers/ PhilippSeuringMasterThesis.pdf.

Valtchev, P., D. Grosser, C. Roume, et M. R. Hacene (2003). GALICIA : an open platform for lattices. In A. d. M. B. Ganter (Ed.), *Using Conceptual Structures : Contributions to ICCS'03*, Aachen (DE), pp. 241–254. Shaker Verlag.

Summary

Relational Concept Analysis (RCA) allows abstractions to be discovered in various software artefacts : Java code or UML models for example. A Model Driven Engineering (MDE) approach applied to RCA leads to a generic abstraction process : it makes possible to build abstractions for any input artefact (UML or Java for example) simply by tuning the approach for the chosen type of input artefact. Until now, this approach has been applied to one-level artefacts (mainly class models). We propose here a study of the application to two-level artefacts, mixing elements and meta-elements (mix of classes and instances, or associations and links for example). This study is applied to a Description Logics, for which we have developped an extension of the RCA-MDE tool.