A Model-driven Engineering Based RCA Process for Bi-level Models Elements / Meta-elements: Application to Description Logics

X. Dolques[1], J.-R. Falleri[1], M. Huchard[1], and C. Nebut[1]

LIRMM, CNRS and Université de Montpellier 2,
161, rue Ada, 34392 Montpellier cedex 5, France
{dolques, falleri, huchard, nebut}@lirmm.fr

Abstract. Relational Concept Analysis (RCA) facilitates the discovery of new abstractions in data descriptions including relations. A model driven approach for RCA implementation makes possible to deal with most input data (models) simply by configuring the transformation for the chosen input data type (metamodel). Until now, we only applied this approach to one-level models (mainly class models). In this paper we study RCA applied to bi-levels models, which mix elements and meta-elements (class-instance models, e.g. OWL models). We propose a model hybridisation approach to tackle the encoding problems and we provide a case study showing the results obtained on OWL models.

1 Introduction

Programs and models are easier to understand and maintain when they are organised using abstractions. Relational Concept Analysis (RCA) is one of the existing approaches to automatically detect such abstractions. RCA is an extension of Formal Concept Analysis (FCA) taking into account the relations linking the analysed entities.

To apply RCA, the analysed entities have first to be encoded into contexts containing information on the attributes of the entities and on the relations linking the entities. Then RCA is applied and builds concept lattices containing the discovered entities, that are then decoded towards the initial language for the entities.

A model-driven engineering (MDE) based approach has been proposed [1,2] to provide a generic mecanism for the encoding and decoding part of the process, that has just to be configured to be adapted to a given language. In this approach, to apply RCA to a given model m, two inputs are needed (in addition to m): the metamodel for m (that can be seen as the structural definition of the language in which m is written), and the configuration making precise which meta-elements of this metamodel have to be taken into account during the RCA process (for example: names of elements, roles of associations, etc).

Until now, such an RCA-MDE process has been successfully applied to class models. The contribution of this paper is to study its application to bi-level models where entities and meta-entities co-exist [3]. Such models are frequently found in cases when we want to represent in the same model a concept and an instance of it, for example in UML instance diagrams [4], RDFS resources [5] or ODM ontologies [6]. We focus in this paper on applying RCA-MDE to individuals in the sense of description logics. The idea is thus to look for abstractions among individuals, each individual being typed by a class. The main issue is to deal with two levels of abstraction: the level of individuals (classically named M0) and the one of classes (M1). The presence of two levels complexifies the application of the RCA-MDE process, as we will show in this paper. After providing background on our approach, we detail two ways to apply RCA-MDE on bi-level models: a naive one, directly inspired from mono-level models, and a more elaborated one giving more relevant results, and that is based on an automated hybridation of the input model and metamodel. We explain how this process is implemented in our RCA-MDE platform, and provide results on two real-world ontologies.

2 Background : Relational Concept Analysis and description logics

Relational Concept Analysis (RCA) Relational Concept Analysis [7] is one of the extensions of Formal Concept Analysis [8] that considers links between objects in the concept construction. Connections can be made with other FCA-based proposals for dealing with relational descriptions or complex structures including [9,10,11,12] to mention just a few. RCA uses a natural representation of data in the form of tables that constitute a relational context family. Some of these tables represent objects of several categories described by binary attributes (*formal contexts*) while the other tables represent relations between objects from the categories (*relational contexts*). We illustrate RCA with an example including a single formal context (K_{nature}, see Fig.1) and two relational contexts, R_{eat} and R_{live}, shown in Figures 3 and 4.

	Thing	Plant	Place	Animal
berry		X		
mountain			X	
sheep				X
lichen		X		
wolf				X
rabbit				X
bear				X
herb		X		

Fig. 1. Formal context K_{nature}

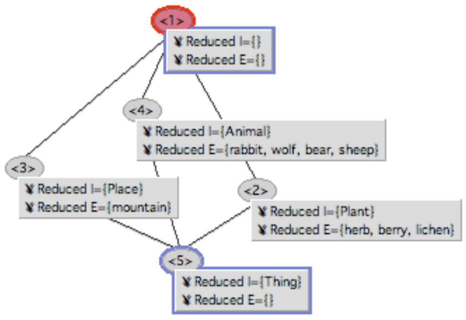

Fig. 2. Concept Lattice L_{nature}

	berry	sheep	lichen	rabbit	herb
berry					
mountain					
sheep					X
lichen					
wolf		X		X	
rabbit					X
bear	X	X	X		
herb					

Fig. 3. Relational context R_{eat}

	mountain
berry	
mountain	
sheep	X
lichen	
wolf	X
rabbit	X
bear	X
herb	

Fig. 4. Relational context R_{live}

Definition 1 (Relational Context Family (RCF)). *A Relational Context Family $\mathcal{R}$ is a pair (K, R). K is a set of formal contexts $K_i = (O_i, A_i, I_i)$, R is a set of relational contexts $R_j = (O_k, O_l, I_j)$ (O_k and O_l are the object sets of the contexts K_k and K_l of K).*

New abstractions emerge iterating the two following steps. The first step is classical concept lattice construction. In second step, formal contexts are concatenated with relational contexts enhanced by concepts created in previous lattice construction.

Initialisation step. Lattices are built at this step using FCA. For each formal context K_i, a lattice $\mathcal{L}_i^0$ is created (in our example, it is shown in Fig. 2).

Step n+1. For each relational context $R_j = (O_k, O_l, I_j)$, an enhanced relational context $R_j^s = (O_k, A, I)$ is created. A is the concept set of the lattice $\mathcal{L}_l^n$ (created at step n). In the case of R_{eat}, we obtain $R_{eats}^s = (O_{nature}, L_{nature}, I)$. I contains the set of pairs (o, a) s.t. $S(R(o), Extent(a))$ is *true*, where S is a *scaling* operator. We consider here two scaling operators: $S_\exists(R(o), Extent(a))$, which is *true* iff $\exists x \in R(o), x \in Extent(a)$, and $S_{\forall\exists}(R(o), Extent(a))$, which is *true* iff $\forall x \in R(o), x \in Extent(a) \wedge \exists x \in R(o), x \in Extent(a)$.

We give a first example using $S_\exists$ to compute R_{eat}^s. Initialisation step allows us to discover the abstraction represented by the concept C_2 (plants). As we have $(berry) \in R_{eat}(bear)$ with $berry \in Extent(C_2)$, $(bear, C_2) \in I$. For similar reasons, $(rabbit, C_2) \in I$. This highlights the fact that bears and rabbits eat at least one kind of plant.

Now we examine a computation based on the scaling operator $S_{\forall\exists}$. As $(sheep) \in R_{eat}(bear)$ and $sheep \notin Extent(C_2)$, now $(bear, C_2) \notin I$. Reversely, since $R_{eat}(rabbit) = \{herb\} \subseteq Extent(C_2)$, we still have $(rabbit, C_2) \in I$. This indicates that rabbits only eat plants, while bears do not only eat plants: but also sheep.

Applying FCA to $K_k \cup \{R_j^s = (O_k, A, I)\}$ creates new concepts that are added to $\mathcal{L}_k^n$ to obtain $\mathcal{L}_k^{n+1}$. For example, still using the scaling operator $S_{\forall\exists}$ on the concatenation of K_{nature}, R_{eat}^s and R_{live}^s (Fig. 5), we obtain the concept

$(\{sheep, rabbit\}, \{Animal, eat : C1, eat : C2, live : C1, live : C3\})$. This concept represents objects that are animals, eat only plants and live only in places (here mountain due to the very restricted example). As the process goes on, more and more complex information on relational structuring emerges. The process stops when lattices at step n are equivalent to those at step $n-1$ *i.e.* when no new concept appear.

	K_{nature}				R^s_{eat}					R^s_{live}				
	Thing	Plant	Place	Animal	C_1	C_2	C_3	C_4	C_5	C_1	C_2	C_3	C_4	C_5
berry		X												
mountain			X											
sheep				X	X	X				X		X		
lichen		X												
wolf				X	X			X		X		X		
rabbit				X	X	X				X		X		
bear				X	X					X		X		
herb		X												

Fig. 5. Context K_{nature} concatenated with enhanced relational contexts R^s_{eat} and R^s_{live}.

Description logics Description logics [13] allow knowledge representation with *Concepts*, *Individuals* and *Roles*. Concepts, included in the terminological box (or TBox), are primitive ones (like *Plant*, *Animal*), constants ($\top$, $\bot$) or defined using several constructors, such as negation ($\neg$), disjunction ($\sqcup$), or conjunction ($\sqcap$). Here we are especially interested in constructors composed with roles: universal role quantification ($\forall R.C$, where R is a role and C is a concept) and existential quantification ($\exists R.C$). $\mathcal{FL}^{-}\mathcal{E}$ is the description logic we will consider in the paper. If *eat* is a role, the concept $Herbivorous$ can be defined with expression $Herbivorous := Animal \sqcap \exists eat.\top \sqcap \forall eat.Plant$, since an herbivorous is an animal which eats at least one thing and which only eats plants. Assertion box (or ABox) contains instanciations. An individual (for example *herb*) is defined by its type (for example $Plant(herb)$), which is a concept of the TBox, while a role is defined by a set of individual pairs like $eat(rabbit, herb)$.

3 Adapting RCA-MDE to bi-level models

RCA in a model-driven engineering approach Lessons learned from previous prototypes [14,15,16] highlighted the need to easily encode data from a large range of models (UML class models in several UML versions, component models, description logics, etc.) into relational context families as well as to parameterise RCA application. The RCA-MDE approach proposes a generic solution to these issues [1,2]. An overview is given in Figure 6.

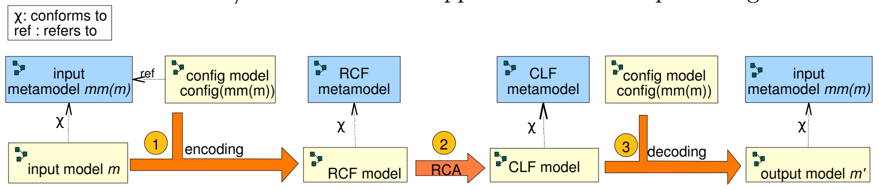

Fig. 6. RCA-MDE approach

To apply RCA to a model m, one just need to provide the corresponding metamodel *mm(m)* and the RCA configuration *config(mm(m))* for this metamodel. *config(mm(m))* defines the meta-elements of *mm(m)* which are considered for analysis. Converting the model into a relational context family is done in two steps. In the first step, a formal context is created for each meta-class indicated in *config(mm(m))*. Binary attributes are the meta-class attributes mentioned in *config(mm(m))*. A specialization/generalization link can be specified for a given meta-class. This link allows to compute the inherited relational attributes. In the second step, a relational context is created for every meta-relation indicated in *config(mm(m))*.

Until now, RCA-MDE has been applied to models owning entities of a single level, i.e. that do not mix entities and meta-entities. In this paper, we show how to apply RCA-MDE on bi-levels models. Such models can be found in models representing instances like ODM [6] that allows to represent ontologies, or UML instance diagrams [4]. We illustrate the approach with description logics : we aim at refactoring models owning individuals based on classes. Those models are composed of a TBox and an ABox (TAB models). We show in this section why a naive adaption based on the one applied for mono-level models is surprisingly not well-suited, and propose an original way to correctly apply it.

Naive adaptation, based on the mono-level modeling To apply RCA to a TAB model, the first (naive) adaptation consists in providing to RCA-MDE a metamodel where coexist: classes, indivuals, relations, and instances of relations, as illustrated at the right of Figure 7. Note that we work with a simplified metamodel of description logics. The model of animals is thus an instance of this metamodel, an excerpt is given at the left of Figure 7, where we only see the animals that live in the moutain.

We also need to provide a configuration model for this metamodel. All the entities of the metamodel (*Class*, *Individual*, *Instance of Relation* and *Relation*) correspond to a formal context. The inter-entity links give rise to relational contexts, and for each relational context, the scaling operator is chosen and defined in the configuration model. We take into account the following associations:

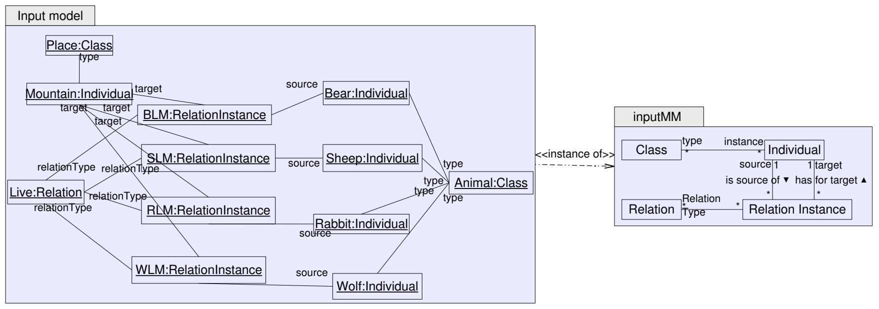

Fig. 7. Naive adaptation for the model of animals.

	ir0	ir1	ir2	ir3	ir4	ir5	ir6	ir7	ir8	ir9	ir10
berry											
mountain											
sheep	X	X									
lichen											
wolf			X	X	X						
rabbit						X	X				
bear								X	X	X	X
herb											

	berry	mountain	sheep	lichen	wolf	rabbit	bear	herb
ir0								X
ir1		X						
ir2						X		
ir3			X					
ir4		X						
ir5								X
ir6		X						
ir7				X				
ir8	X							
ir9			X					
ir10		X						

	eat	live
ir0	X	
ir1		X
ir2	X	
ir3	X	
ir4		X
ir5	X	
ir6		X
ir7	X	
ir8	X	
ir9	X	
ir10		X

Fig. 8. Relational contexts of relations *is source of* (lhs), *has for target* (center) and *relation type* (rhs).

– *type*[1] linking *Individual* to *Class*. The associated scaling operator is $S_{\exists}$. The relational context R_{type} will thus be created, associated to the $S_{\exists}$ operator;
– *is source of* linking *Individual* to *Relation Instance* (it leads to generate the $R_{issourceof}$ context). We chose here the $S_{\forall\exists}$ operator;
– *has for target* linking *Relation Instance* to *Individual* (it leads to generate the $R_{hasfortarget}$ context). We chose here the $S_{\exists}$ operator;
– *relation type* linking *Relation Instance* to *Relation* (it leads to generate the $R_{typerelation}$ context). We chose here the $S_{\exists}$ operator.

We thus have a single formal context for all the instances of relations. The relation context *relation type* represents the links between those instances of relations and the relations (see Figure 8). Yet if abstractions can be found with this configuration, others cannot, that could however be discovered applying RCA in a classical way, i.e. filling the contexts without taking into account the way input data are modeled. The problems arise when using a scaling operator different from $S_{\exists}$. For example, let us refer to animals linked to their habitation and their diet. Using RCA, we hope obvious abstractions to appear with operator $S_{\forall\exists}$, e.g. herbivorous (animals such that, whatever they eat, it is plant), carnivorous (animals such that whatever they eat, it is animal) and omnivorous (animals eating both vegetal and animal food). However, all the relation instances (links) are

[1] *type* is in fact a role of this association.

represented by the same metaclass in the metamodel and they belong to a single formal context. As a first consequence we cannot apply different scaling operators to the relations (e.g. $S_{\forall\exists}$ for *eat* and $S_{\exists}$ for *live*). As a second consequence, some concepts built by original RCA (on relational context family like in Section 2) cannot be found. While original RCA builds a concept including *sheep* and *rabbit* (because all *eat* links end at *herb* which is included in C_2 extent), naive modeling of RCA-MDE cannot (because all links ends are not included in a non trivial concept extent : *eat* links go towards *herb* while *live* links go towards *mountain*).

The source of the problem comes from the following causes. First, our approach, due to a genericity matter, creates the formal contexts from the elements of a metamodel. Second, we use a model with two levels of abstraction, and the instanciation relation allowing to go from one level to the other is defined in the metamodel by a simple association. In our case, it clearly appears that an instanciation relation exists between the relations and the instances of relations, but the relations do not belong to the metamodel, thus it is not possible to create a formal context for each relation. We thus propose in the next section a more complex yet more adequate solution.

Adaptation for bi-levels models: hybridisation of the input metamodel with the input model We propose to *promote* a part of the model, i.e. to move the relations in the metamodel, in the form of relation of model, and to transform the relation *relation type* into an instanciation relation. We thus hybridise a part of the input model with a part of the input metamodel, as illustrated in Figure 9. The input model is then also modified as shown in the right of Figure 9. The hybridisation transformation aims at deleting the reification of the relations in a model (by the concept *Instance of relation*) since it is not relevant to look for new abstractions.

The configuration model follows the same idea as the transformation: we only take into account two entities of the metamodel: *Class* and *Individual*, and the following relations:

- *type* linking *Individual* to *Class*. We associate it the scaling operator $S_{\exists}$. The relational context R_{type} will thus be created, associated to the operator $S_{\exists}$;
- *live* linking *Individual* to *Individual*, with a scaling operator (e.g. $S_{\forall\exists}$).
- *eat* linking *Individual* to *Individual*, with a scaling operator (e.g. $S_{\forall\exists}$).

This solution does not imply to modify the RCA-MDE process, we only modify input models. Those modifications result in a relational context of each relation of the input model (in our example: exactly the contexts of Figures 3 and 4). Reading the tables is easier, because the instances of relations are represented by a table per relation, and this table owns the source and the target of the instance of relation.

Hybridising the model has then for consequence to obtain back the model elements leveling while still keeping the same information on the input model ; and the relations of the input model that were semantically relations on instances become actual relations on instances. The hybrid model with its metamodel is then a mono-level model on which the RCA-MDE process can be applied in a

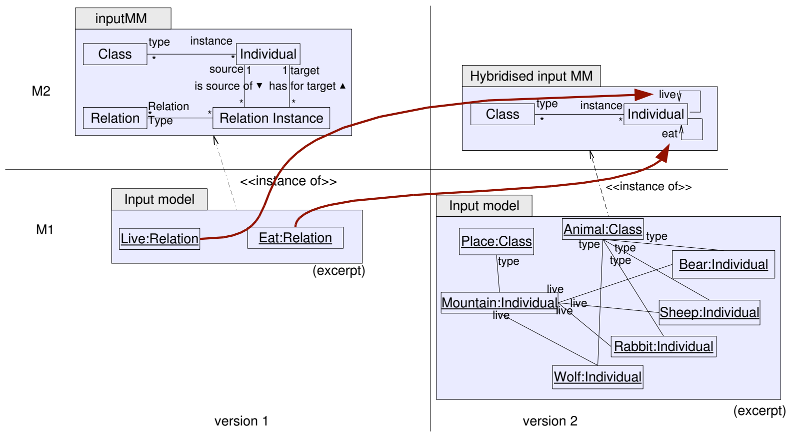

Fig. 9. Hybridisation of the metamodel.

classical way. The new definitions of concepts in the context of the individuals will depend on a single relation. In our example, we will be able to define a concept to which *sheep* belongs and which gathers all the individuals that only eat plants, independently from the other types of relations like *live*. This modeling corresponds to what can be obtained with a classical application of RCA.

4 Platform and experimentations

Platform The platform implementing RCA-MDE uses the modeling framework EMF [17]. It is based on the meta-modeling language Ecore to read models and their metamodels.

To represent description logics models, we use the OWL language (Web Ontology Language, [18]). OWL is intensively used in Web technologies as knowledge representation language, and many modeling tools exist for OWL, as well as many OWL models. Our metamodel is included, renaming the entities, in the one proposed in the Eclipse plugin [19] that allows to handle OWL models with EMF. In this way, a TAB model can easily be translated into an OWL model, renaming `Class` into `OWLClass`, etc.

To adapt the RCA-MDE platform to TAB models, keeping in mind that we want the process to remain generic (adaptable to other input data), we have to define the hybridisation transformation of the OWL metamodel for a given TAB model (see Figure 10), and a configuration making explicit which meta-elements have to be taken into account by the RCA. The hybridisation transformation is fully automated, it is written in Java and uses EMF to handle models. It takes as input the OWL metamodel, a TAB model and the configuration model and generates both the hybridised metamodel, the new TAB model conform to the

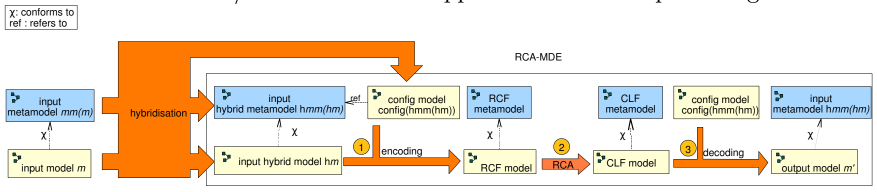

Fig. 10. Illustration of the metamodel hybridisation in RCA-MDE.

hybridised metamodel, and the corresponding configuration. As shown in Figure 9, the hybridisation transformation lists all the relations and puts them up a level of abstraction higher, so that they appear in the hybridised metamodel. This transformation only modifies the *relations* and the *instances of relation*. The transformation also impacts the configuration model, since it refers to a given metamodel. The metamodel being hybridised, the configuration must also be hybridised. The obtained configuration can then be modified by the final user, in particular for the choice of the scaling operators.

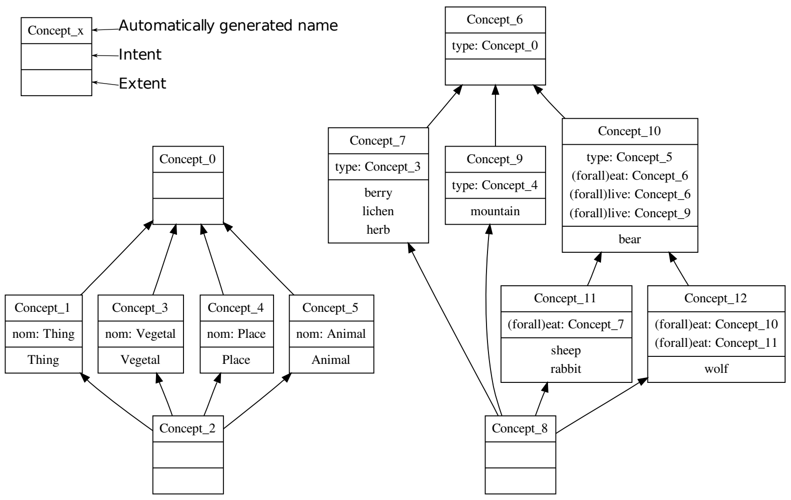

Fig. 11. Lattices obtained applying RCA-MDE on the example of animals. The class lattice is on the lhs, the Individual lattice on the rhs.

Figure 11 shows, using Hasse diagrams, the lattices obtained applying RCA-MDE on the example of animals. The lattice we focus on is the one obtained from the context of the *Individuals*. We see that concepts have been created to group the elements of same type: *Concept_ 7* groups the elements of type *vegetal*, *Concept_ 9* groups those of type *place* and *Concept_ 10* groups those of type *animal*. *Concept_ 11* groups animals that eat only vegetals. *Concept_ 12* groups the animals that eat only animals, in particular animals of *Concept_ 11*. From the intent of the new concepts, we can infer in an automatic way a logical formula defining them.

The platform resulting from this work allows us to analyse bi-level models in an automatic way: as well as for the analysis of mono-level models, we just need to configure the RCA with a configuration model, so we kept the generic approach. Our experiments showed us that the configuration for bi-level models is slightly more complex since it frequently handles several scaling operators.

Table 1. Models studied. The Ontology *Model* is about surface water and water quality model , the Ontology *Autos* is about Automobiles and their equipment. We take into account the number of *Individuals*, *Classes*, *Object Properties* and their instances we called *Links* in this table. We compare the number of Concepts obtained using FCA and RCA in the lattice from *Individual* Context.

Ontology name	Individuals	Classes	Object Properties	Links	FCA concepts	RCA concepts
Model[2]	114	20	8	273	31	65
Autos[3]	321	91	13	375	82	175

We have tested our approach of hybridisation on some real ontologies in OWL format (Tab.1). We compare the result[4] obtained using Formal Concept Analysis (we take the results obtained after the first iteration of the RCA process) to those obtained using Relational Concept Analysis. In the following we will concentrate on the lattice from *Individual* context, as the lattice from the *Class* lattice cannot produce new concepts (there is no relation in the metamodel with *Class* elements as source).

On the *Model* Ontology, FCA would generate 31 concepts. Nearly each class is transformed in one concept, except for 3 classes which have the same extent. Our approach generated 55 concepts using a $S_\exists$ operator of scaling and 65 concepts using a $S_\exists$ and a $S_{\forall\exists}$ operator of scaling. From the concepts obtained by FCA, 7 of them have a more precise intent with RCA.

We note that concept intents here describe the presence of relation between individuals more than a value of the relation. This is caused by the fact that a lot of classes do not have subclasses.

The use of $S_{\forall\exists}$ scaling in addition with $S_\exists$ scaling gives us a way to know the most specialized type for the target of a relation. To deduce the type of a relation target, you need to know the target type of all the instances of this relation.

[2] http://loki.cae.drexel.edu/˜wbs/ontology/model.htm

[3] http://gaia.isti.cnr.it/˜straccia/Teaching/IS/2007/Exercises/autos.owl

[4] all the lattices can be found at http://www.lirmm.fr/˜dolques/publications/data/cla08

The RCA approach applied on bi-level models produces a different type of result from applied on mono-level. On bi-level models we take a set of elements, and the process produces new subsets depending on the properties of the elements. On the examples we studied in OWL, the partitioning depends on the type of the individuals, and the Object Property (name of relations in OWL vocabulary) of which they are sources. For instance FCA can produce a concept that groups the individuals of type *ModelingSystem* and that are linked by the Object Property *hasModel* to another individual. RCA shows us that the individual of this concept are all linked by the ObjectProperty *hasModel* to Individuals of type *Model* which all have an Individual of type *Organisation* as a developer (concept `NMWithAv_Dev_Org_ModelDim`).

This kind of result could help to complete under-specified ontologies in an approach by-example, by showing which kind of data is missing in individual description and by restricting the domain and range of an Object Property.

5 Conclusion

Until now, building abstractions using an RCA process and an MDE paradigm has been applied to models with a single level of abstraction, i.e. models that do not mix entities and meta-entities. In this paper, we have studied how to use an RCA-MDE approach for bi-level models, where co-exist meta-entities and their instances. We based our work on the example of description logics: we focused on models mixing individuals and links between individuals with the classes typing the individuals and the relations defining the links. We have shown that a direct application of the approach does not give the expected results, and proposed a more complex solution giving results similar to those obtained with an application of RCA with a manual encoding of data. This solution is based on the promotion of the instances of relations from the input model up to the input metamodel.

In this paper, we worked with description logics with minimal expressivity; in particular we do not take into account language elements existing in OWL like the specialization relation that could be applied to classes, or the constraints that could be added on the sources and targets of the relations. Future work will consist in dealing with more complex description logics, as well as in integrating the obtained results in the original model. We also plan to make the hybridisation generic instead of specific to our description logics metamodel, in order to validate our approach with other kinds of bi-level models such as UML models with classes and instances.

References

1. Arévalo, G., Falleri, J.R., Huchard, M., Nebut, C.: Building abstractions in class models: Formal concept analysis in a model-driven approach. In: Proc. of the MoDELS'06 conference. (2006) 513–527
2. Falleri, J.R., Huchard, M., Nebut, C., Arévalo, G.: Use of Model Driven Engineering in Building Generic FCA/RCA Tools. In Diatta, J., Eklund, P., Liquière,

M., eds.: Proc. of CLA'07: Fifth International Conference on Concept Lattices and Their Applications. (2007) 225–236
3. Dolques, X., Falleri, J.R., Huchard, M., Nebut, C.: Adaptation d'un processus de construction d'abstractions basé idm à des modèles bi-niveaux éléments / méta-éléments : Application aux logiques de description. In: LMO Conference. (2008)
4. OMG: UML 2.0 superstructure. Technical report, Object Management Group (2004)
5. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C (2004)
6. OMG: Ontology definition metamodel. Technical report, Object Management Group (2007)
7. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. **49**(1-4) (2007) 39–76
8. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin (1999)
9. Priss, U.: Classification of meronymy by methods of relational concept analysis. In: Online Proceedings of the 1996 Midwest Artificial Intelligence Conf., Bloomington, Indiana. (1996)
10. Prediger, S., Wille, R.: The lattice of concept graphs of a relationally scaled context. In: Proc. of the 7th Intl. Conf. on Conceptual Structures (ICCS'99), Springer (1999) 401–414
11. Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In Delugach, H., Stumme, G., eds.: Conceptual Structures: Broadening the Base, Proc. of the 9th Intl. Conf. on Conceptual Structures (ICCS'01), Stanford, CA. Volume 2120 of LNCS., Springer (2001) 129–142
12. Ferré, S., Ridoux, O., Sigonneau, B.: Arbitrary relations in formal concept analysis and logical information systems. In Dau, F., Mugnier, M.L., Stumme, G., eds.: ICCS. Volume 3596 of Lecture Notes in Computer Science., Springer (2005) 166–180
13. Baader, F., Nutt, W.: Basic description logics. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003) 43–95
14. Valtchev, P., Grosser, D., Roume, C., Hacene, M.R.: Galicia: an open platform for lattices. In B. Ganter, A.d.M., ed.: Using Conceptual Structures: Contributions to ICCS'03, Aachen (DE), Shaker Verlag (2003) 241–254
15. Dao, M., Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Towards Practical Tools for Mining Abstractions in UML Models. In Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J., eds.: ICEIS (3). (2006) 276–283
16. Seuring, P.: Design and implementation of a UML model refactoring tool. Master's thesis, Hasso-Plattner-Institute for Software Systems Engineering at the University of Postdam (2005) http://www.lirmm.fr/~huchard/Documents/Papiers/PhilippSeuringMasterThesis.pdf.
17. Budinsky, F., Grose, T., Steinberg, D., Ellersick, R., Merks, E., Brodsky, S.: Eclipse Modeling Framework: a developer's guide. Addison-Wesley Professional (2003)
18. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. Technical report, W3C (2004)
19. EODM: site du projet EODM, http://www.eclipse.org/modeling/mdt/?project=eodm