Merging Conceptual Hierarchies Using Concept Lattices

Amine Mohamed Rouane Hacene, Petko Valtchev, Houari Sahraoui, Marianne Huchard

Merging conceptual hierarchies using concept lattices

Mohamed H. Rouane[a] and Petko Valtchev[a] and Houari Sahraoui[a] and Marianne Huchard[b]

[a] DIRO, Université de Montréal, C.P. 6128, Succ. "Centre-Ville", Montréal, Canada, H3C 3J7;
[b] LIRMM, UMR 5506, 161 rue Ada, 34392 Montpellier cedex 5, France

ABSTRACT

In many situations, one faces the need for integrating a set of conceptual hierarchies that represent parts of the same domain. The target structure of the integration is a unique conceptual hierarchy that embeds at least the total of the knowledge encoded in the initial ones. The key issues to address here are the discovery of higher-level abstractions on top of the existing concepts and the resolution of naming conflicts among these. Formal concepts analysis (FCA) is a mathematical approach toward abstracting from attribute-based object descriptions which has been recently extended to fit relational descriptions thus giving rise to the relational concept analysis (RCA) framework. Building up on RCA, our integration approach amounts to encoding the initial hierarchies into set of binary tables, one for each component category, e.g., classes, associations, etc., and subsequently constructing, querying and reorganizing the corresponding abstraction hierarchies until a unique satisfactory hierarchy is obtained. This paper puts the emphasis on the mechanisms of discovering new abstractions and exporting them between the abstraction hierarchies of related component categories. The impact of naming conflicts within the RCA process is discussed as well. The paper uses UML as description language for conceptual hierarchies.

Keywords: Hierarchy integration, formal concept analysis, abstraction, naming conflict resolution.

1. MOTIVATION

Specialization hierarchies embody the knowledge about a particular domain expressed by means of concepts, concept features and inter-concept relationships (general kind relationships, composition, aggregation, etc.). The design of such hierarchies is known to be a hard-to-automate problem, even in the cases where the ground set of objects/classes is given and the goal is simply to organize them into a meaningful structure. Consequently, most methods that support the hierarchy design task apply some inductive techniques in a semi-automated or interactive mode.

We focus here on the integration of specialization hierarchies. In fact, the need for merging two or more existing specialization hierarchies into a unique global one that encodes at least the initial hierarchical knowledge occurs in many practical situations. Thus, in database design, it is known as *schema integration*,[1] in knowledge engineering as *ontology merge*,[2] in software modeling as *model integration*[3](e.g., assembly of subjects or aspect inter-weaving), etc. Whatever the name given and the domain-specific constraints, the integration of hierarchies basically consists in assembling a set of potentially incomplete views on the same reality.

Yet the assembly is more than a mere juxtaposition of hierarchies as parts of these may overlap whereas overlapping between concepts of different hierarchies may remain partial, thus suggesting these are the variants of a more general concept. To sum up, integration requires the detection of overlapping parts of the initial hierarchies whereby some of the corresponding elements, i.e., concepts, relationships, properties, etc., stemming from different hierarchies will be directly recognized as identical whereas, others, although similar, will rather give rise to abstractions that have not been discovered beforehand.

In this paper, we discuss the automated support for the integration process which remains manual in its very nature. Indeed, identity recognition for elements of different hierarchies is ultimately up to the designer's judgment as concept naming is prone to errors and ambiguity, e.g., the same domain element may be given different, yet synonymous, names in different hierarchies. Moreover, the choice of the "interesting" abstractions among the set of all plausible ones is hardly automated. We sketch an approach for hierarchy integration that relies on abstraction mechanisms from the formal concept analysis (FCA)[4] field. FCA turns a binary (individuals

(rouanehm,valtchev,sahraouh)@iro.umontreal.ca, huchard@lirmm.fr

x properties) table into a complete lattice made up of pairs of closed sets. To cover more sophisticated concept descriptions, a version of FCA is used that feeds inter-individual relations into the generalization process. In our framework, mechanisms detecting regularities in individual/property co-occurrences and in the lattice structure are relied on to spot potential naming conflicts.

2. INTEGRATION CHALLENGES

In the remainder of the paper, we consider hierarchies described in UML.[5] Using the underlying terminology, *abstractions* represent generic domain concepts and are obtained by factorizing the shared specifications of classes or associations.

Figure 1 shows two class diagrams pertaining to strict subsets of the banking domain, i.e., the bank account sub-domain (henceforth identified by BA) and credit card sub-domain (CCA). Please notice that there is a substantial overlap between both in terms of business classes, associations and class members.

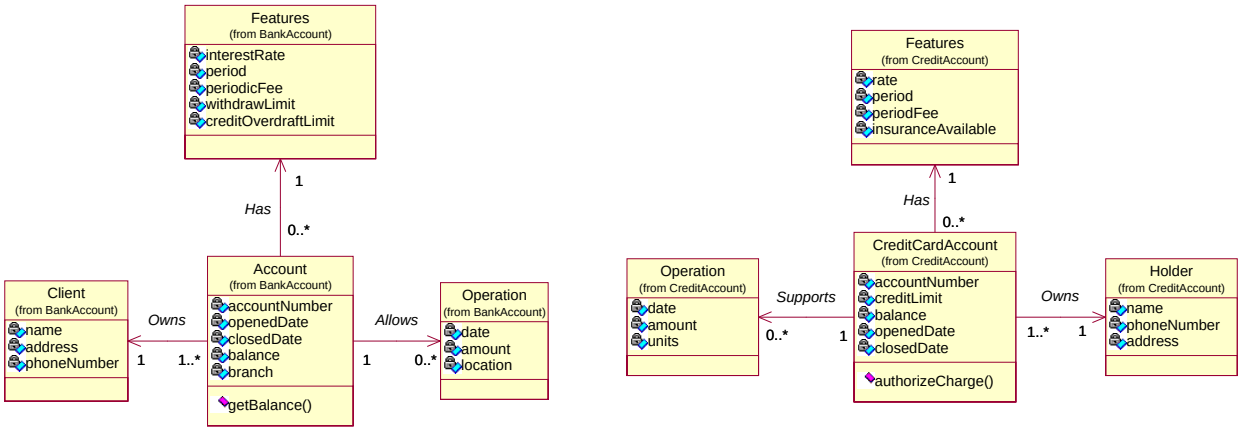

Figure 1. Left: Bank account sub-domain model. **Right:** Credit card sub-domain model.

When only a sub-domain is focused on, then obviously, the possibilities of identifying the appropriate generic concepts are limited. In particular, concepts transcending the sub-domain boundaries would be overlooked. For instance, although both hierarchies in Figure 1 are faithful representations of the underlying sub-domains, none of them identifies the general concept of account that would factor the specifications of `CreditCardAccount` and (check) `Account` (see class `Account` in Figure 6). These can only be discovered when the appropriate sub-domain hierarchies are put side-by-side.

Consequently, a clever integration approach should look for all the 'hidden' domain abstractions that cross the local hierarchy boundaries. As we shall show it in the following paragraphs, mathematical tools exist that, provided an appropriate encoding is performed beforehand, deliver the set of *all potentially useful abstractions* over a set of concrete entities.

There is yet another difficulty integration has to face which is due to the overlapping of some hierarchies. Indeed, restricted views and insufficient synchronization may cause mismatches in the naming of domain elements that happen to appear in several sub-domains, as the case of the `Client`/`Holder` classes suggests. Indeed, both `Client` and (card) `Holder` seem to reflect the same reality, i.e., bank customers whose only differentiation, except for naming, lays in the sort of banking products they possess. Hence, and despite the diverging class names, these classes should be merged in the global hierarchy (see Figure 6). In contrast, the pair of `Feature` classes, although identically named, rather translate variations of a hypothetic *Feature* entity. Again, the name (mis)match is due to the limited view in both local hierarchies: the lack of intra-hierarchy variation prevented the identification of the generic class `Feature` whose absence prompted its local variants (subclasses) to be named identically. It is noteworthy that in both above cases of misleading naming, the specification of the compared entities, i.e,, the list of attributes, is a strong indicator of whether these are identical or not: while both `Feature` classes share

only a subset of their properties, the description of `Client` and `Holder` match perfectly. In fact, integration is also a reconciliation activity on two inherently incomplete, i.e., partial, views on the same reality.

In the next section we present a domain that studies the construction of conceptual hierarchies from observation from a mathematical point of view and therefore provides a formal framework for the abstraction activity that is considered here.

3. FCA-BASED INTEGRATION

3.1. FCA basics

A context $\mathcal{K} = (O, A, I)$ is given where O is a finite set of formal objects, A is a finite set of formal attributes and $I \subseteq O \times A$ is a binary relation between these sets, saying that the object $o \in O$ has the attribute $a \in A$ whenever $(o, a) \in I$ holds.

	accountNumber	address	amount	balance	branch	closeDate	creditLimit	creditOverdraftLimit	date	insurance	interestRate	location	name	openedDate	period	periodicFee	phoneNumber	units	withdrawLimit
BA-Account	X			X		X								X					
Client		X											X				X		
BA-Features								X			X				X	X			X
BA-Operation			X						X			X							
CCA-Account	X			X		X	X							X					
CCA-Operation			X						X									X	
CCA-Features										X	X				X	X			
Holder		X											X				X		

Table 1. Binary context $\mathcal{K}^0_{class}$. Formal objects are UML classes and formal attributes are their variables.

Table 1 represents a context drawn from Figure 1. The image of an object set $X \in O$ is defined by: $X' = \{a \in A \mid \forall o \in O : (o, a) \in I\}$ and dually the image of an attribute set $Y \in A$ is defined by: $Y' = \{o \in O \mid \forall a \in A : (o, a) \in I\}$. For example within Table 1, $\{\texttt{BA-Account}, \texttt{CCA-Account}\}' = \{\texttt{balance}\}$ and $\{\texttt{withDrawLimit}\}' = \{\texttt{BA-Features}, \texttt{CCA-Features}\}$. A (formal) concept in $\mathcal{K}$ is a pair $(X, Y) \in \mathcal{P}(O) \times \mathcal{P}(A)$ for which $X = Y'$ and $Y = X'$ where X is called the *extent* and Y the *intent*. The set $\mathcal{C}$ of all concepts provided with an order relation based on extent inclusion, say $\mathcal{L}$, is a complete lattice. The lattice drawn from the context of Table 1 is shown in Figure 2 on the right. As many practical applications involve non-binary data, *many-valued contexts* have been introduced in FCA. In such a context $\mathcal{K} = (O, A, V, J)$, an object o is described by a set of attribute value pairs (a, v), meaning that J is a ternary relation that binds the objects from O, the attributes from A and the values from V (see table presented on the left-hand side of Figure 2). The construction of a lattice for $\mathcal{K}$ requires a pre-processing step, called *conceptual scaling*,[4] that derives a binary context out of $\mathcal{K}$.

3.2. FCA-based hierarchy analysis

FCA has been successfully applied to class hierarchy design and maintenance[6] with contexts drawn from sets of existing classes by encoding classes and class members as formal objects and attributes, respectively. The resulting conceptual lattice was interpreted as a surrogate hierarchy where formal concepts represent (abstract) software classes and order in the lattice as specialization.

Recently, Huchard *et al.*[7] proposed an extension of the basic FCA mechanisms, called *relational concept analysis* (RCA), that allows links among formal objects to be fed into the lattice construction in order to abstract from those links higher order inter-concept relations similar to UML associations. RCA uses a compound data

	out-assoc	in-assoc
BA-Account	{Allows, BA-Owns, BA-Has}	
BA-Features		BA-Has
BA-Operation		Allows
CCA-Features		CCA-Has
CCA-Operation		Supports
Client		BA-Owns
CCA-Account	{CCA-Owns, CCA-Has, Supports}	
Holder		CCA-Owns

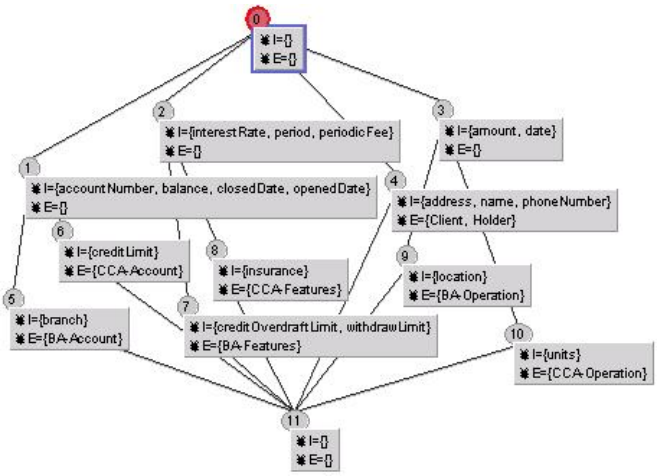

Figure 2. Left: Relational variables in $\mathcal{K}_{class}$. **Right:** The lattice $\mathcal{L}^0_{class}$.

format, called *relational context family* (RCF), including a set of formal contexts and a set of inter-object relations (translated to *relational* attributes for computational reasons). Each context, called *relational context*, describes a separate sort of domain entities as formal objects, while relations connect entities from (possibly) separate contexts. An encoding of a UML class diagram maps classes and associations to separate contexts of the RCF and the class - association relations to dedicated relational attributes in both contexts (e.g., `in-association`, `target-class`, etc.). An innovative feature of RCA is the inference of relational abstractions connecting formal concepts from links between formal objects, following a tight analogy to the Entity-Relationship data model. In our UML encoding, this means that the intent of a formal concept over the class context may include a relational attribute "pointing at" a formal concept of the associations context, which abstracts from existing links between the corresponding formal objects of class and association type.

The production of inter-concept relations is integrated into the overall lattice construction step by means of *relational scaling* mechanism.[8] RCA uses an iterative lattice construction mechanism that processes all the contexts of a family simultaneously. The construction alternates relational scaling and lattice construction tasks. Thus, each lattice gets more detailed along subsequent iterations (since new concepts are added while the old ones remain) whereas the global construction halts whenever a particular step yields no new concept.

Figure 2, on the left, presents a relational context extracted from the UML diagrams given in Figure 1 in a way that will be presented later on. RCA uses an iterative lattice construction method that processes all the contexts of a family simultaneously by means of *relational scaling* mechanism which encodes relational attributes into binary ones. Figure 3 presents the lattice of the context obtained by concatenating Table 1 and table shown in the left hand-side of Figure 2.

3.3. Naming conflicts resolution

A set of hierarchical elements may overlap either on their *names* or on their *descriptions* (including properties and links to other elements). Both these aspects of an element are to be seen as "proxies" for the denotation behind the element, i.e., the domain entity it represents, which is not necessarily available for a hypothetical automated tool. Indeed, although providing some rigor and a lot of structure, plain UML models, as other hierarchy description languages, are basically natural-language-bound and therefore potentially ambiguous. Obviously, a strong match in both name and description is a good indication for elements representing the same reality. Overlap in only one aspect is trickier case as it may be the sign, except for imprecise modeling, of complex linguistics phenomena intervening such as *synonymous* (same entity, different names, e.g., `Client`/`Holder`) or *polysemous* (same name, different entities, e.g., the `Feature` classes).

These problems have been addressed in the database schema integration field and it is usually suggested that name conflicts be resolved through name normalization based on linguistic resources such as electronic

dictionaries or specialized domain ontologies. Of course, neither is the linguistic-knowledge based approach universally applicable, nor is it possible to solve all conflicts by an automated tool. Therefore, a more realistic agenda for an integration tool could be to detect the suspect areas in the global hierarchy where effective/potential name conflicts reside. FCA offers mechanisms for the detection of typical and exceptional patterns of co-occurrence of formal objects/attributes called implication rules.[9] These can be used to detect discrepancies between descriptions of elements, i.e., mainly classes and associations, having same or close names that may hide a conflict. Structural properties of the lattice and mutual position of those elements within it could be further explored to the same end.

More sophisticated tools could even suggest, for a given pair of elements, the impact of a particular action on them, e.g., merge, creation of a dedicated generalization, split into thinner elements, etc., to the remaining lattice and hence the final hierarchy. Finally, a more extensive and fully manual approach towards name conflict resolution could be based on the *attribute exploration*[10] mechanism which is basically a knowledge acquisition method assisted by a automated tool asking questions to the expert designer. The tool guides the hierarchy designer to the discovery of a minimal set of implications that represent the entire domain while minimizing the number of questions. It is noteworthy that the above mentioned FCA tools such as implication extraction and attribute exploration are yet to be extended to the relational data descriptions that are used here.

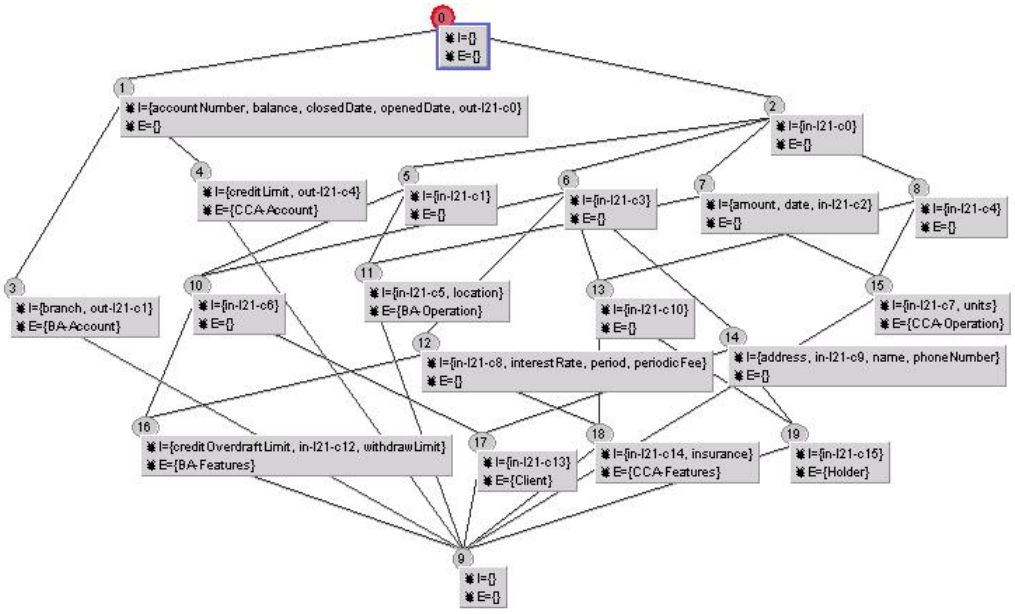

Figure 3. Relational lattice $\mathcal{L}^1_{class}$.

3.4. Description of the approach

The global scenario of integration includes three main stages: *encoding*, *abstraction* and *reverse encoding*. During the encoding stage, the initial hierarchies are transformed into a unique RCF where each context correspond to a sort of objects in the UML meta-model, i.e., classes, associations, methods and variables. Here we limit our considerations to classes and associations. Moreover, the incidences between meta-objects are translated into relational attributes in the RCF, e.g., the attribute `target-class` in Table 2. The encoding of the entire running example is jointly represented by Tables 1 and 2, together with the left-hand side of Figure 2. The resolution of some naming conflicts takes place at this step, in particular conflicts in attribute/method names, which are often less ambiguous than the class or association ones.

At the abstraction stage, a set of inter-related concept hierarchies are constructed on top of the global RCF.[11] The global hierarchy construction starts by processing only the non-relational part of context attributes. Next,

	name	s-mult	t-mult	nav	source-class	target-class
Allows	allows	1	0..N	st	BA-Account	BA-Operation
BA-Owns	ba-owns	1..N	1	st	BA-Account	Client
BA-Has	ba-has	0..N	1	st	BA-Account	BA-Features
CCA-Owns	cca-owns	1..N	1	st	CCA-Account	Holder
CCA-Has	cca-has	0..N	1	st	CCA-Account	CCA-Features
Supports	supports	1	0..N	st	CCA-Account	CCA-Operation

Table 2. Context $\mathcal{K}_{assoc}$ of associations.

the process iterates between construction and scaling of relational attributes in the RCF until a fixed point is reached, i.e., isomorphism between lattices at step $i+1$ and their counterparts at step i. Relational scaling turns each object-valued attribute into a set of binary ones each of whom represents a concept from the conceptual hierarchy of the underlying context. For a relational attribute α that is scaled along an existing lattice on its range context, the resulting scale attributes will have the form α-l#j#i-c#k where j is the context number (here 1 stands for classes and 2 for associations), i refers to the step of the iterative construction process and k is the concept index in $\mathcal{L}^i_j$. For instance, as the values of target-class in $\mathcal{K}_{assoc}$ are objects in $\mathcal{K}_{class}$ (see Table 2), the attributes of the scale context $\mathcal{K}_{target-class}$ correspond to concepts from $\mathcal{L}^0_{class}$ (see Figure 4 on the left). They are used to compute the relational extension of $\mathcal{K}^1_{assoc}$ along the link target-class (see the tc-l10-cX attribute in Figure 4 on the right). Figures 3 and 5 show the final hierarchy derived from the class and association contexts, respectively. Reverse encoding turns the resulting lattices into a global UML class

	tc-l10-c0	tc-l10-c2	tc-l10-c3	tc-l10-c4	tc-l10-c7	tc-l10-c8	tc-l10-c9	tc-l10-c10
BA-Features	X	X			X			
BA-Operation	X		X				X	
CCA-Features	X	X				X		
CCA-Operation	X		X					X
Client	X			X				
Holder	X			X				

	tc-l10-c0	tc-l10-c2	tc-l10-c3	tc-l10-c4	tc-l10-c7	tc-l10-c8	tc-l10-c9	tc-l10-c10
Allows	X		X				X	
BA-Owns	X			X				
BA-Has	X	X			X			
CCA-Owns	X			X				
CCA-Has	X	X				X		
Supports	X		X					X

Figure 4. Left: Scale context $\mathcal{K}_{target-class}$. **Right:** Relational extension of $\mathcal{K}^1_{assoc}$ through $\mathcal{K}_{target-class}$.

diagram (see Figure 6). The process starts with an initial set of class concepts and examines iteratively both lattices following inter-concept relational attributes to figure out the appropriate associations. Initial class set includes all the object-concepts which represent existing local classes and some of the attribute-concepts in $\mathcal{L}^n_{class}$ which means the $\mathcal{GSH}$ of $\mathcal{L}^n_{class}$ is the focus. However, concepts with exclusively relational scale attributes in their intents are left aside at this step as they represent potential classes of high generality level. In our running example, the object-concepts in $\mathcal{L}^1_{class}$ are as follows (corresponding classes in Figure 6 given in brackets): #3 (BA-Account), #16 (BA-Features), #11 (BA-Transaction), #4 (CCA-Account), #18 (CCA-Features), #15 (CCA-Transaction), #17, #19. The concepts #17 and #19 are discarded as they hold only links in their intents. The attribute-concepts in $\mathcal{L}^1_{class}$ are also considered: #1 (Account), #7 (Transaction), #12 (Feature), and #14 (Holder) are kept while the remaining concepts such as #2 and #5 are ignored. The selected concepts, once assigned a name each, constitute a first draft of the global UML model. The next step consists in detecting the appropriate associations from the relational information in the retained class concepts. According to our forward encoding, concepts from $\mathcal{K}_{class}$ have incoming and outgoing links to concepts from $\mathcal{K}_{assoc}$. Moreover, as these links play symmetrical role, tracking one of them is enough. The scale attributes of the incoming link in each selected concept (prefix in) are examined and only those corresponding to a minimum of the related concept set are kept for association computation. For example, the concept #7 in $\mathcal{L}^1_{class}$ which corresponds to the class Transaction has two such attributes in-l21-c0 and in-l21-c2 hence potentially two incoming associations.

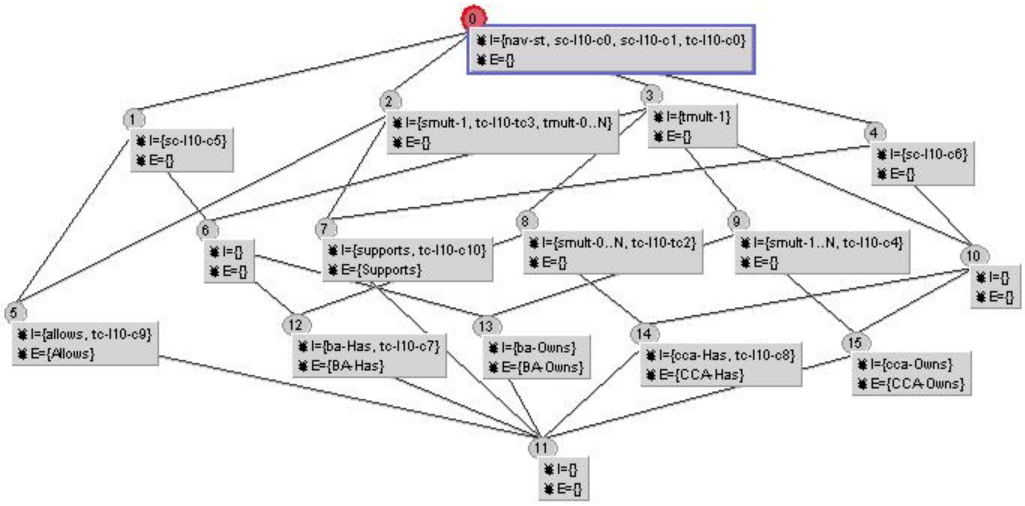

Figure 5. Relational lattice $\mathcal{L}^1_{assoc}$.

However, the concepts #0 (`in-l21-c0`) is a parent of #2 (`in-l21-c2`) in $\mathcal{L}^1_{assoc}$ and thus the unique incoming association for `Transaction` is described by #2. The latter has two source class attributes, `sc-l10-c0` and `sc-l10-c1`, whereby the corresponding concepts in $\mathcal{L}^0_{class}$, i.e., #0 and #1, satisfy #1 $\leq$ #0. Thus, the source class is #1 which corresponds to the super-concept of the object-concepts for both account classes, i.e., the class `Account` in our interpretation. As the target of the new association is known to be `Transaction`, the exploration of the relations between $\mathcal{L}^1_{class}$ and $\mathcal{L}^1_{assoc}$ continues with another class concept. Figure 6 shows the result of the integration process. Clearly, interesting abstractions have been discovered: `Account` generalizes both sorts of accounts and `Transaction` the respective operations on them, `Holder` become the generic account owner after discarding the `Owner` class as independent one. A new associations have been found as well, i.e., `Supports` which links `Transaction` and `Account`.

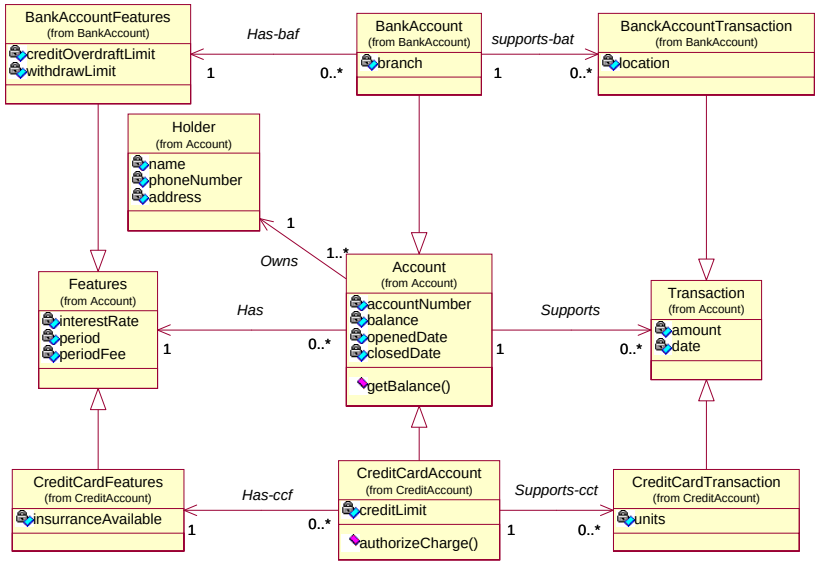

Figure 6. Global UML model.

4. RESEARCH AVENUES

We addressed the integration of a set of UML class diagrams into a global one as an instance of the generic hierarchy integration problem. The underlying challenges amount to abstracting new and previously unknown concepts and properly match overlapping parts of the hierarchies. We suggested a RCA-based framework for integration where concepts and relations are organized in a separate but mutually related abstraction hierarchies, the concept lattices. The core abstraction process is highly complex while the gap between a conceptual hierarchy and its encoding in terms of a RCF requires an effective set of tools that facilitate the back and forth navigation.

Our focus in future steps will be on increasing the precision of the mapping from a hierarchy to a RCF and on the design of effective tools for naming conflict detection and on-line adaptive restructuring of the hierarchies. A possible approach would be to assess term "similarity" using a lexical database as in.[12] Moreover, we shall look at interactive simplification (pruning) of the obtained hierarchies and at increasing the automation of the identification of relevant classes and associations as starting point for the reverse-encoding of the global hierarchy. Finally, studying practical cases to assess the quality of the integrated hierarchies.

REFERENCES

1. I. Mirbel and J.-L. Cavarero, "An integration method for design schemas," in *Conference on Advanced Information Systems Engineering*, pp. 457–475, 1996.
2. N. Noy and M. Musen, "Promptdiff: A fixed-point algorithm for comparing ontology versions," in *Proc. 18th AAAI*, 2002.
3. H. Ossher and P. Tarr, "Using multidimensional separation of concerns to (re)shape evolving software," *Commun. ACM* **44**(10), pp. 43–50, 2001.
4. B. Ganter and R. Wille, *Formal Concept Analysis, Mathematical Foundations*, Springer, Berlin, 1999.
5. Object Management Group, Inc., *OMG Unified Modeling Language Specification*, version 1.5 ed., March 2003.
6. R. Godin, H. Mili, G. Mineau, R. Missaoui, A. Arfi, and T. Chau, "Design of Class Hierarchies Based on Concept (Galois) Lattices," *Theory and Practice of Object Systems* **4**(2), 1998.
7. M. Huchard, C. Roume, and P. Valtchev, "When concepts point at other concepts: the case of uml diagram reconstruction," in *Proceedings of the 2nd Workshop on Advances in Formal Concept Analysis for Knowledge Discovery in Databases (FCAKDD)*, pp. 32–43, 2002.
8. P. Valtchev, M. H. Rouane, M. Huchard, and C. Roume, "Extracting Formal Concepts out of Relational Data," in *Proceedings of the 4th Intl. Conference Journées de l'Informatique Messine (JIM'03): Knowledge Discovery and Discrete Mathematics, Metz (FR), 3-6 September*, E. SanJuan, A. Berry, A. Sigayret, and A. Napoli, eds., pp. 37–49, INRIA, 2003.
9. J. Guigues and V. Duquenne, "Familles minimales d'implications informatives résultant d'un tableau de données binaires," *Mathématiques et Sciences Humaines* **95**, pp. 5–18, 1986.
10. B. Ganter, "Attribute exploration with background knowledge," **217**, pp. 215–233, 1999.
11. M. Dao, M. Huchard, M. H. Rouane, C. Roume, and P. Valtchev, "Improving generalization level in uml models: Iterative cross generalization in practice," in *Proceedings of the 12th International Conference on Conceptual Structures (ICCS'04)*, **14**, Springer-Verlag, LNCS 3127, July 2004.
12. M. Lafourcade and V.Prince, "Relative synonymy and conceptual vectors," in *Proceedings of NLPRS2001, Tokyo*, pp. 127–134, 2001.