An environment for building and maintaining class hierarchies

Hervé Leblanc, Christophe Dony, Marianne Huchard, Thérèse Libourel
LIRMM,
161 rue Ada, 34392 Montpellier Cedex 5, France,
email: ename@lirmm.fr

30th April 1999

Abstract

This paper presents a project the goal of wich is to design and implement a high level environment for building and maintaining (reorganizing, merging, etc) class hierarchies having formal properties. These properties are ensured by the Galois sub-hierarchy underlying structure used to constraint them. This project is connected with an industrial project, named PÉTRA, in which telecommunications applications have been developped.

We have specified an extension of the UML meta-model able to represent any kind of hierarchies we want to deal with, including those developed in the PÉTRA project which have an original layered model.

We then have specified a set of basic and higher-level operations required in such an environment to deal with hierarchies.

1 Introduction

A crucial problem of object oriented software development is the production and maintenance of "good" class hierarchies. We consider these as examples of object-oriented software architecture. One of the criteria used to evaluate class hierarchies is properties factorization. The Galois lattice is a mathematical structure allowing to constraint class hierarchies having the maximal factorization property. The interactive environment we want to build is devoted to the controlled manipulation of thses hierarchies. The following sections describe the main steps of the project : the specification of a UML meta-model to encode hierarchies, the specification of the relevant set of basic operations to manipulate hierarchies, what will be used either directly by users or to build highler level operations such reorganization or merging ones. Two paradigms governs our approach: first we propose an help dedicate to manage "step after step" evolution of class hierarchies, and second we propose tools for global reorganization or merge of these.

2 Main aspects of the project

2.1 A new UML Meta-model

Firstly, we have defined a meta-model to represent hierarchies. It is designed to deal with heterogeneous hierarchies coming from either design tools (OMT, UML) or languages (C++, JAVA). It is a subset of UML kernel (*Types, Classes, Interfaces, Attributes, Methods, Associations,* ...), extended to deal with PÉTRA object model particularities and to include features useful to manage hierarchy evolution and building.

Our UML extension is able to deal with the following specific features.

- *relevant* classes i.e. abstract classes that the designer does not want to be removed.
- *generic* properties, introduced to manage overloading and overriding. A *generic* property is a set of properties (attributes or methods), partially ordered by a specialization or subtyping relation.
- *layers*. The PÉTRA object model is based on a layered architecture. A layer is a hierarchy (an instance of our own meta-model) whose components (classes, interfaces, associations) have a common semantics and a same abstraction level [8]. Layers are themselves connected by a dependence relation. The dependence between layers are only inheritance links [7].

2.2 Galois lattice

The Galois lattice is a mathematical structure used in several domains such as machine learning and knowledge representation. The interest of such a structure as a model to represent inheritance hierarchies has been hightlighted in [10]. Given a binary relation associating a set of objects together with the properties they own, the Galois lattice gives all maximal objects subsets sharing properties and symmetrically all maximal property subsets belonging to several objects, all these subsets being ordered by inclusion. The "Galois subhierarchy" is a particular part of the Galois lattice containing the more useful information, which can be viewed as an inheritance hierarchy in which properties are maximally factorized. All of these concepts are represented in figure 1.

We discuss in the following the main drawbacks and advantages of Galois lattice and Galois sub-hierarchy.

drawbacks

- The Galois lattice is a complex structure (exponential in objects and properties number). Simplifications made to obtain the Galois subhierarchy reduce this complexity which becomes polynomial.
- This structure prescribes multiple inheritance, in languages which do not have this feature, additional adjustments have to be done.
- Furthermore a question remains : is the maximal factorization always the better choice in terms of object design ?

advantages

- This structure is formally defined and provides a good and easy characterization of the operations and algorithms results.
- The maximal factorization ensured by the construction induces a maximal reuse of properties. Each property being declared in a single class, the scope of a change can be easily located.
- The structure seems quite complex, but it is the minimal one ensuring a maximal factorization.

- Thanks to the structure, some of the algorithms have good properties, for example, using the incremental algorithm to build a hierarchy adding the classes successively gives a result that does not depend on the order the classes are added.

- Lastly, researchers which have proposed various algorithms to globally build hierarchies, have used implicitely [5, 6, 12] or explicitely [9, 10] the same or very similar structures.

2.3 Platform Operations

We propose an environment in which all relevant construction and evolution operations on hierarchies preserve the Galois subhierarchy structure.

Up to now, the only algorithms of this kind are "global algorithms" [10] which use the binary relation (object-property) to build the hierarchy, and " incremental algorithms" [9] which add a class to a hierarchy. Other approaches, outside of the Galois lattice model, build a hierarchy [2] or add a new class [3] or present a consistant set of elementary operations having only a local impact (for example in the context of object-oriented databases [1]).

We propose the following operations, each of them maintaining a Galois subhierarchy structure.

First, there are already known algorithms for the following operation. We consider this operation as initial one (for non empty hierarchies).

- global hierarchy construction, from a binary relation mapping the objects to their properties. All necessary factorization classes are added;

Second, there are incremental operations that are used to modify a hierarchy. We call them evolution operations.

- addition of a class : the algorithm ARES [9], for example, can be used "incremental algorithm";

- deletion of a class C from a Galois subhierarchy;

- property addition to a single class C;

- property addition to a subset of a hierarchy;

- deletion of a property p from a class C;

- deletion of a property p from of a hierarchy;

- inheritance link addition, the consequence is the addition of properties inherited through this link;

- inheritance link deletion, this may be interpreted as the deletion of properties inherited through this link.

Third, there are more general operations.

- fusion: given any two hierarchies, a new hierarchy integrating them is built.

- prune: given a hierarchy, take a subpart of this;

3 Conclusion

Our contribution is twofold: we have defined a meta-model encoding many heterogeneous hierarchies of classes and we have introduced operations that act on a more basic representation of these hierarchies.

Futhermore, our aim is to be as close as possible of a realistic object model in the implementation of operations, as described in our meta-model [4, 11]. The following problems have to be considered.

- dealing with property masking and overloading (cf. meta-model);
- integrating and using the concept of "interface" as in JAVA ;
- dealing with associations.

Finally, we will have to address the problem of the evolution of a component already involved in "client" programs. Evolution should neither remove nor modify the component features. Some new questions are then raised :

- What are the conditions ensuring a good reuse ?
- What preconditions have to be given for the evolution operations ?
- On the basis of the previous operations, what interesting compound operations may be proposed ?

References

[1] J. Banerjee, W. Kim, K.J. Kim, and H. Korth. Semantics and implementation of schema evolution object-oriented databases. *Proc. ACM SIGMOD Conf.*, 1987.

[2] P. Bergstein. Object Preserving Class Transformations. *Special issue of Sigplan Notice - Proceedings of ACM OOPSLA'91*, 26(11):299–313, 1991.

[3] E. Casais. Automatic reorganization of object-oriented hierarchies. *Object-Oriented Systems*, 1(2):95–115, 1994.

[4] C.Dony, M.Huchard, H.Leblanc, and T.Libourel. Méta-modèle de représentation de hiérarchies de classes. Technical report, CNET, Avril 1998. Rapport d'avancement 01.98, pages 2-29.

[5] J.-B. Chen and S. C. Lee. Generation and reorganization of subtype hierarchies. *Journal of Object Oriented Programming*, 8(8), 1996.

[6] William R. Cook. Interfaces and Specifications for the Smalltalk-80 Collection Classes. *Special issue of Sigplan Notice - Proceedings of ACM OOPSLA'92.*, 10(1):1–15, 1992.

[7] Michel Dao. Evolution of layered object-oriented models. Submited to WECOOP'99.

[8] Pétra: Groupe de travail. Modèle générique de réseaux commutés pour l'ingénierie du trafic. Technical report, CNET, Septembre 1997. DE/DAC/GTR/29.97.

[9] H. Dicky, C. Dony, M. Huchard, and T. Libourel. On automatic class insertion with overloading. *Special issue of Sigplan Notice - Proceedings of ACM OOPSLA'96.*, 31(10):251–267, 1996.

[10] R. Godin and H. Mili. Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices. *Special issue of Sigplan Notice - Proceedings of ACM OOPSLA'93*, 28(10):394–410, 1993.

[11] H.Leblanc. Définition formelle des opérations élémentaires. Technical report, CNET, Décembre 1998. Rapport d'avancement 02.98.

[12] Ivan Moore and Tim Clement. A Simple and Efficient Algorithm for Inferring Inheritance Hierarchies. *TOOLS Europe 1996 Proceedings, Prentice-Hall*, 1996.

	a	b	c	d	e	f
o1	x	x	x			
o2			x			
o3	x		x	x		x
o4	x		x		x	x
o5	x	x				

properties(o1) = { a, b, c }
properties(o2) = { c }
properties(o3) = { a, c, d, f }
properties(o4) = { a, c, e, f }
properties(o5) = { a,b }

In Input, a binary relation R on E = {o1, o2 , o3, o4, o5} and F = {a, b, c, d, e, f}

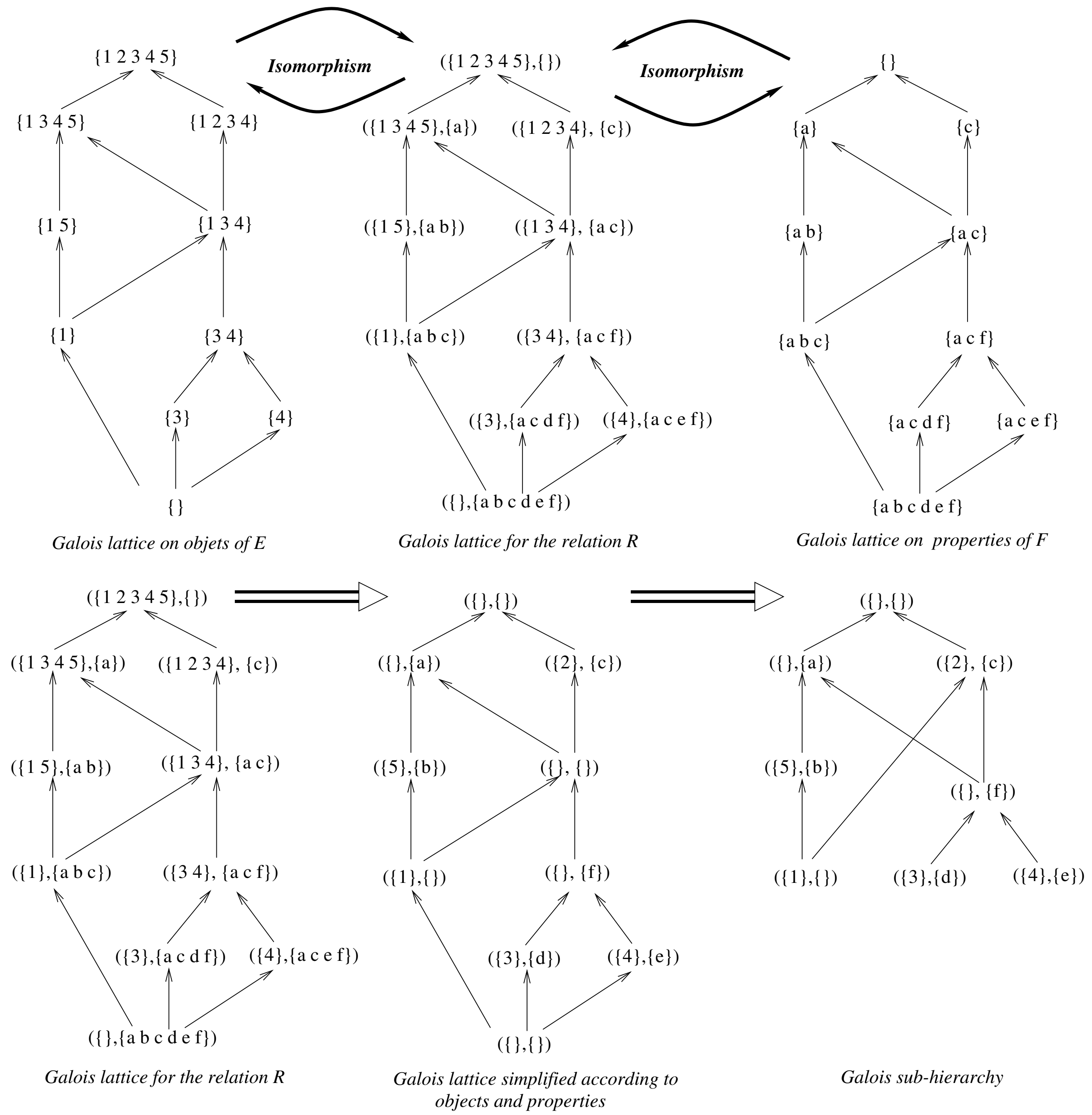

Figure 1: Galois lattice and sub-hierarchy