Variability representation in product lines using concept lattices: feasibility study with descriptions from Wikipedia's product comparison matrices

Jessie Carbonnel, Marianne Huchard, and Alain Gutierrez

LIRMM, CNRS & Université de Montpellier, France
jessiecarbonnel@gmail.com
surname.lastname@lirmm.fr

Abstract. Several formalisms can be used to express variability in a product line. Product comparison matrix is a common and simple way to display variability of existing products from a same family, but they lack of formalisation. In this paper, we focus on concept lattices, another alternative already explored in several works to express variability. We first propose a method to translate a description from existing product comparison matrices into a concept lattice using Formal Concept Analysis. Then, we propose an approach to represent the case where a product family is described by other product families with interconnected lattices using Relational Concept Analysis. Because of the combinatorial aspect of these approaches, we evaluate the scalability of the produced structures. We show that a particular structure (AOC-poset) possesses interesting properties for the studies that we envision.

Keywords: Product lines, Formal Concept Analysis, Variability Representation.

1 Introduction

In product line engineering [5], several formalisms can be used to depict variability. Variability representation usually requires to take into account a large amount of data, and it is important to provide tools to help designers to represent and exploit them.

Among existing formalisms, the most common is Product Comparison Matrices (PCMs). PCMs describe product properties in a tabular form. It is a simple way to display features of products from a same family and to compare them. However, there is no existing format or good practices to design these PCMs. Therefore, cells in PCMs lack of formalisation and it is difficult to perform automatic and efficient processing or analysis on them [4, 17]. Feature Models (FMs) constitute an alternative to PCMs. FMs describe a set of existing features and constraints between them, and thus represent all possible configurations of products from a same family [6, 9, 11, 12]. They depict variability in a more formal

Local Proceedings also appeared in ISBN 978-84-606-7410-8, Universidad de Málaga (Dept. Matemática Aplicada), Spain.

way than PCMs, but, in their stardard form, FMs focus exclusively on features and do not specify if an existing product is associated with a configuration.

Formal Concept Analysis (FCA) and concept lattices have already been studied to express variability [13]. From a set of objects described by attributes, FCA computes subsets of objects that have common attributes and structures these subsets in a hierarchical way in concept lattices. Concept lattices can represent in a more formal way than PCMs informations related to variability. Through their structure, concept lattices highlight constraints between attributes like FMs, while keeping the relation between existing products of the family and the possible configurations. Besides, contrarily to FMs, which can have many various forms depending design choices, concept lattices represent variability in a canonical form. Moreover, FCA can be extended by Relational Concept Analysis (RCA) which permits to take into account relationships between objects of separate lattices and provide a set of interconnected lattices. This is useful to classify sets of products from different categories.

Concept lattices formal and structural aspects make them good candidates to apply automatic or manual processing including product comparison, research by attributes, partial visualisation around points of interest, or decision support. But the exponential growth of the size of concept lattices can make them difficult to exploit. In this paper, we will study the dimensions of these structures and try to find out if depicting product line variability with concept lattices provides structures that can be exploited from a perspective of size. For this, we will build in a first phase concept lattices from existing descriptions. Because there are abundant and focus on both products and features, we will extract descriptions from PCMs. Besides, we can find PCMs that possess a feature whose value domain corresponds to a set of products described by another PCM. It is a special case where a product family is described by another product family. In a second phase, we will model this case by interconnecting concept lattices obtained from the descriptions of these two PCMs using Relational Concept Analysis.

In this paper, we want to answer these questions: How can we represent the variability expressed by a PCM with a concept lattice? To what extent can we model the case where a product family is described by another product family with RCA? Can we efficiently exploit structures obtained with FCA and RCA with regard to their size?

The remainder of this paper is organised as follows. In Section 2, we will review Formal Concept Analysis and propose a way to build a concept lattice from a PCM's description. In Section 3, we will study how to represent the case where a product family is described by another product family with Relational Concept Analysis. Then, in Section 4, we will apply these two methods on Wikipedia's PCMs to get an order of magnitude of the obtained structure size. Section 5 discusses related work. Section 6 presents conclusion and future work.

2 Formal Concept Analysis and product comparison matrices

This section proposes an approach to represent with a concept lattice the variability originally expressed in a PCM.

2.1 Concept lattices and AOC-posets

Formal Concept Analysis [8] is a mathematical framework which structures a set of objects described by attributes and highlights the differences and similarities between these objects. FCA extracts from a formal context a set of concepts that forms a partial order provided with a lattice structure: the concept lattice.

A formal context is a 3-tuple (O, A, R) where O and A are two sets and $R \subseteq O \times A$ a binary relation. Elements from O are called *objects* and elements from A are called *attributes*. A pair from R states that *the object o possesses the attribute a*. Given a formal context $K = (O, A, R)$, a concept is a tuple (E, I) such that $E \subseteq O$ and $I \subseteq A$. It depicts a maximal set of objects that share a maximal set of common attributes. $E = \{o \in O | \forall a \in I, (o, a) \in R\}$ is the concept's extent and $I = \{a \in A | \forall o \in E, (o, a) \in R\}$ is the concept's intent. Given a formal context $K = (O, A, R)$ and two concepts $C_1 = (E_1, I_1)$ and $C_2 = (E_2, I_2)$ from K, $C_1 \leq C_2$ if and only if $E_1 \subseteq E_2$ and $I_2 \subseteq I_1$. C_1 is a subconcept of C_2 and C_2 is a superconcept of C_1. When we provide all the concepts from K with the specialisation order $\leq$, we obtain a lattice structure called a concept lattice.

We represent intent and extent of a concept in an optimised way by making elements appear only in the concept where they are introduced. Figure 7 represents a concept lattice having simplified intent and extent. We call object-concept and attribute-concept the concepts which introduce respectively at least an object or an attribute. In Figure 7, *Concept_7* and *Concept_2* introduce neither attributes nor objects: their simplified intent and extent are empty. If they are not necessary, we can choose to ignore these concepts. *Attribute-Object-Concept poset* (AOC-poset) from a formal context K is the sub-order of $(C_K, \leq)$ restricted to object-concepts and attribute-concepts. Figure 4 presents the AOC-poset matching the concept lattice from Figure 7. In our case, interesting properties of concept lattices with regard to variability are preserved in AOC-posets: this smaller structure can be used as an alternative to concept lattices.

2.2 Product Comparison Matrix

A PCM describes a set of products from a same family with variable features in a tabular way. Figure 1 presents a PCM which describes four products against two features, taken from Wikipedia.

We notice that the cells of this PCM lack of formalisation: in this, different values have the same meaning (*Object-Oriented* and *OO*) and it seems that there are no rules on the use of value separator (',' or '&' or 'and').

Language	Standardized	Paradigm
Java	Yes	Functional, Imperative, OO
Perl	No	Functional & Procedural & Imperative
Php	No	Functional, Procedural, Imperative and Object-Oriented
C#	Yes	functional, procedural, imperative, Object Oriented

Fig. 1. Excerpt of a Product Comparison Matrix on programming languages (http://en.wikipedia.org/wiki/Comparison_of_programming_languages, July 2014)

2.3 Processing PCMs to get formal context

If we want to automatically build concept lattices from this kind of descriptions, we need to make the cell values respect a format in order to extract and process them. In [17], authors identify height different types of cell values:

- *yes/no values* that indicate if the criterion is satisfied or not,
- *constrained yes/no values* when the criterion is satisfied under conditions,
- *single-value* when the criterion is satisfied using this value,
- *multi-values* when several values can satisfy the criterion,
- *unknown value* when we do not know if the criterion is satisfied,
- *empty cell*,
- *inconsistent value* when the value is not related to the criterion,
- *extra information* when the cell value offers additional informations.

We clean each type of cells as follows. Empty cells are not a problem, even though they indicate a lack of information. Inconsistent values should be detected, then clarified or erased. Unknown values and extra informations will be simply erased. Other types of cells could have either one single value or a list of values. We will always use a coma as value separator. Values with the same meanings will be written in the same way. Once we have applied these rules on a PCM, we consider that this PCM is cleaned. A cleaned PCM can own three types of features:

- *simple boolean*,
- *constrained boolean*,
- *non-boolean*.

Since automatic process is difficult on original PCMs, we clean them manually. When we clean the PCM in Figure 1, we obtain the PCM in Figure 2.

We can now automatically extract values from cleaned PCMs to generate *formal contexts*. Given a set of objects and a set of binary attributes, a formal context is a binary relation that states which attributes are possessed by each object. In summary, we want to convert a set of multivalued features (PCM) into a set of binary attributes (formal context).

Scaling technique [8] consists in creating a binary attribute for each value (or group of values) of a multivalued feature. Boolean features can produce a single attribute to indicate if either or not the object owns this feature. Yet

Language	Standardized	Paradigm
Java	Yes	Functional, Imperative, Object-Oriented
Perl	No	Functional, Procedural, Imperative
Php	No	Functional, Procedural, Imperative, Object-Oriented
C#	Yes	Functional, Procedural, Imperative, Object-Oriented

Fig. 2. PCM in Figure 1 cleaned manually

because of empty cells, the fact that an object does not possess an attribute could also mean that the cell from the PCM was left blank. To be more precise, we can choose to generate two attributes: one to indicate that the object possesses the feature, and one to indicate that the object does not possess the feature. Constrained boolean features can be processed in the same way than simple boolean features by producing one or two attributes, with the difference that we can keep constraints in the form of attributes. Non-boolean features will simply produce an attribute per value or group of values. We applied scaling technique on the cleaned PCM of Figure 2 and got the formal context in Figure 3.

Language	Standardized:Yes	Standardized:No	Functional	Procedural	Imperative	Object -Oriented
Java	x		x		x	x
Perl		x	x	x	x	
Php		x	x	x	x	x
C#	x		x	x	x	x

Fig. 3. Formal context obtained after scaling the cleaned PCM in Figure 2

The structure of concept lattices and AOC-posets permits to highlight interesting properties from variability point of view: they classify objects depending on their attributes and emphasise relations between these attributes (e.g. require, exclude). For instance: attributes introduced in the top concept are owned by all objects; attributes which are introduced in the same concept always appear together; if an object o_1 is introduced in a sub-concept of a concept introducing an object o_2, o_1 possesses all the attributes of o_2 and other attributes; two objects introduced in the same concept possess the same attributes. Feature models show part of this information, mainly reduced to relations between attributes, as they do not include the products in the representation.

Figure 7 represents the concept lattice from the formal context of Figure 3. Figure 4 represents the AOC-poset from the formal context of Figure 3. In these two structures, we can see that: all languages permit to write programs according

to functional and imperative paradigms; the product *Php* has all the attributes of *Perl* in addition to the attribute *Object Oriented*; all standardised languages are object-oriented.

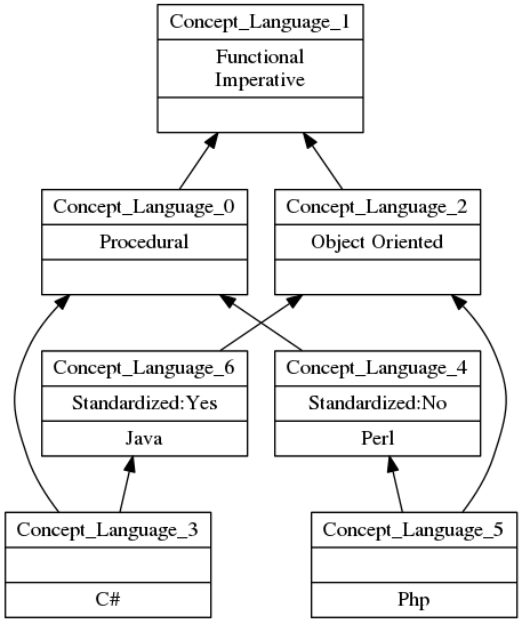


Fig. 4. AOC-poset from the formal context of Figure 3

3 Relational Concept Analysis and interconnected product lattices

This section proposes an approach to model the case where a PCM possesses a feature whose value domain corresponds to a set of products described by another PCM.

3.1 Modeling interconnected families of products with RCA

We illustrate the modeling of interconnected families of products with an extension of our example. We can find on Wikipedia a PCM on *Wikisoftwares* that refers to *Programming Languages*: we want to structure wikisoftwares according to programming languages in which they are written. We assume a PCM about wikisoftwares that owns a boolean feature *Open Source* and a constrained

boolean feature *Spam Prevention*. We applied Section 2 approach and obtained the formal (objects-attributes) context in Figure 5. Figure 8 presents the concept lattice associated with the context in Figure 5.

Wikisoftware	OS:Yes	OS:No	SP:Yes	SP:No	SP:Captcha	SP:Blacklist
TeamPage		x	x		x	
Socialtext			x			
MindTouch		x	x			
DokuWiki	x		x			x
EditMe		x	x		x	

Fig. 5. Objects-attributes context of Wikisoftwares

Relational Concept Analysis (RCA) [10] extends FCA to take into account relations between several sets of objects. Each set of objects is defined by its own attributes (in an objects-attributes context) and can be linked with other sets of objects. A relation between two sets of objects is stated by a relational context (objects-objects context). A relational context is a 3-tuple (O_1, O_2, I) where O_1 (source) and O_2 (target) are two sets of objects such that there are two formal contexts (O_1, A_1, R_1) and (O_2, A_2, R_2), and where $I \subseteq O_1 \times O_2$ is a binary relation.

We want to express the relation *isWritten* between objects of *Wikisoftwares* and objects of *Programming languages*. TeamPage and EditMe are written in Java, SocialText in Perl, DokuWiki in Php and Mindtouch in both Php and C#. We link each wikisoftware with corresponding programming languages in an objects-objects context, and we present it in Figure 6.

isWritten	Java	Perl	Php	C#
TeamPage	x			
Socialtext		x		
MindTouch			x	x
DokuWiki			x	
EditMe	x			

Fig. 6. Objects-objects context expressing the relation between objects of Wikisoftwares and Programming languages

3.2 Processing interconnected families of products

Given an objects-objects context $R_j = (O_k, O_l, I_j)$, there are different ways for an object from O_k domain to be in relation with a concept from O_l. For instance: an object is linked (by I_j) to at least one object of a concept's extent (*existential*

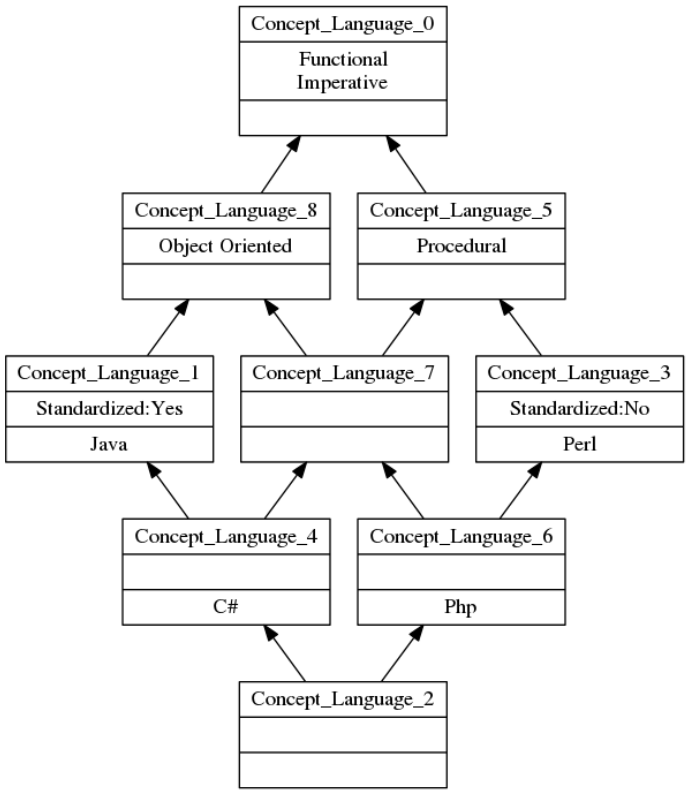


Fig. 7. Concept lattice of Programming languages, step 0 (built with RCAExplore: http://dolques.free.fr/rcaexplore.php)

scaling); an object is linked (by I_j) only to objects of a concept's extent (*universal scaling*). For each relation of R, we specify which scaling operator is used.

In RCA, objects-attributes contexts are extended according to objects-objects contexts to take into account relations between objects of different sets. For each objects-objects context $R_j = (O_k, O_l, I_j)$, RCA extends the objects-attributes context of the set of objects O_k by adding *relational attributes* according to concepts of the lattice associated with the objects-attributes O_l. Each concept c from O_l gives a relational attribute $q\ r : c$ where q is a scaling operator and r is the relation between O_k and O_l. A relational attribute appears in a lattice just as the other attributes, with the difference that it can be considered like a *reference* to a concept from another lattice.

As shown on the example, data are represented in a Relational Context Family (RCF), which is a tuple (K, R) such that K is a set of objects-attributes contexts $K_i = (O_i, A_i, I_i)$, $1 \leq i \leq n$ and R is a set of objects-objects contexts

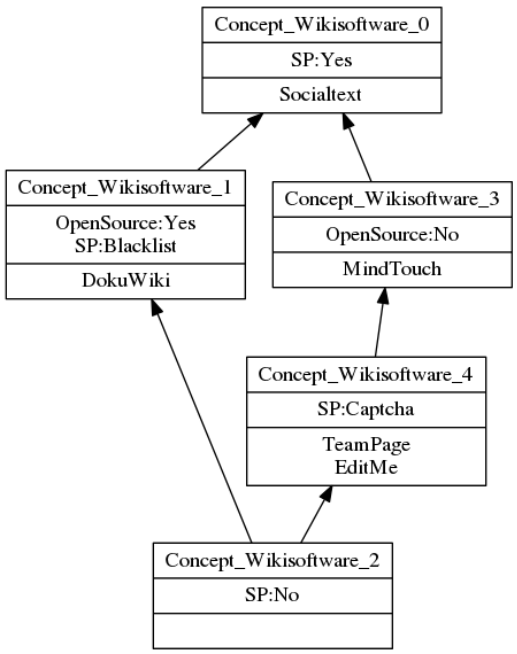


Fig. 8. Concept lattice of Wikisoftwares, step 0

$R_j = (O_k, O_l, I_j)$, $1 \leq j \leq m$, with $I_j \subseteq O_k \times O_l$. Given an objects-attributes context $\mathcal{K} = (O, A, I)$, we define $rel(\mathcal{K})$ the set of relations (objects-objects contexts) of R which have O for domain, and ρ a function which associates a scaling operator to each objects-objects context of R. For each step, we extend the context $\mathcal{K}$ by adding relational attributes from each context of $rel(\mathcal{K})$: we obtain the *complete relational extension of* $\mathcal{K}$. When we compute the complete relational extension of each context of K, we obtain the *complete relational extension of the RCF*.

RCA generates a succession of contexts and lattices associated with the RCF (K, R) and ρ. In step 0, RCA generates lattices associated with contexts of K. $K^0 = K$. In step $e + 1$, RCA computes complete relational extension of the K^e contexts. The obtained extended contexts (K^{e+1}) possess relational attributes which refer to concepts of lattices obtained in the previous step.

In our example, the RCF is composed of *Wikisoftware*, *Programming Language* and of the objects-objects context *isWritten*. When we compute the complete relational extension of this RCF, we extend the objects-attributes context of *Wikisoftware* with relational attributes which refer to each concept of *Programming language*. Figure 7 and Figure 8 represent lattices of *Wikisoftware* and *Programming language* at step 0. Figure 9 presents the concept lattice from the

extended objects-attributes context of *Wikisoftwares*, at step 1. In this example, we cannot go further than step 1.

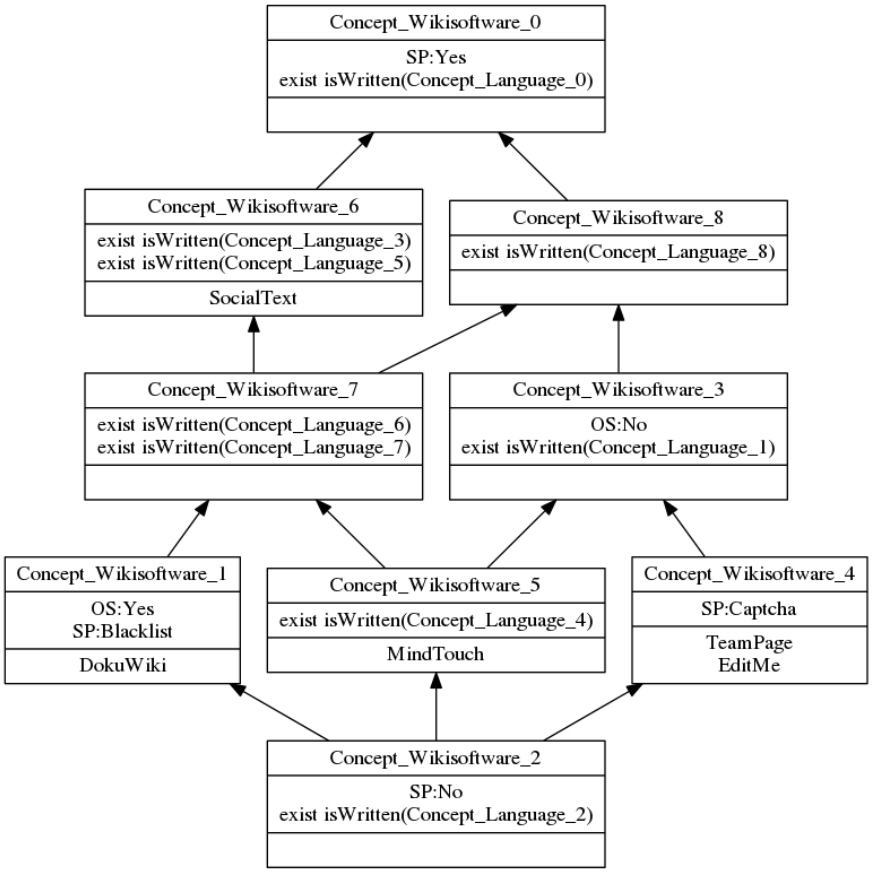


Fig. 9. Concept lattice of Wikisoftwares, step 1

In Figure 9, relational attributes are read like references to concepts in the lattice of *Programming languages* at step 0 (Figure 7). An extended concept lattice gives us the same kind of informations that are emphasised in a basic concept lattice, but it takes into account attributes from other product families. This brings a new dimension to research and classification of products from a same family. In our example, it permits us to select a wikisoftware depending on the paradigm of its programming language. In Figure 9, we can read for instance:

– *DokuWiki* (concept 1) is written in a programming language characterised by concepts 6, 7, 8, 3, 5 and 0 of *Programming languages*, corresponding

to attributes *Standardized:No*, *Procedural*, *Functional*, *Object Oriented* and *Imperative*;
- *Team Page* and *EditMe* are equivalent products because they are introduced in the same concept (same attributes and same relational attributes);
- a wikisoftware not *Open Source* is written in a standardised language;
- an *Open Source* wikisoftware is written in an unstandardised language.

4 Assessing scalability of the approach

Until now, we proposed a first method to obtain a concept lattice from a PCM's description and a second method to depict the case where features possess their own PCM with interconnected lattices. In this section, we evaluate these two methods on existing PCMs from Wikipedia to get an order of magnitude of the obtained structures size.

The number of generated concepts from a formal context depends on the number of objects, the number of attributes and the form of the context: this number can reach $2^{min(|O|,|A|)}$ with a lattice, and $|O| + |A|$ for an AOC-poset.

In the following tests, we generate both concept lattices and AOC-posets to emphasise the impact of deleting concepts which introduce neither attributes nor objects on the size of the structure. Each test was made twice, a first time with *full formal contexts* (scaling technique giving two attributes for each boolean feature, and keeping constraints in the form of attributes for constrained boolean features) and a second time with *reduced fomal contexts* (scaling technique giving one attribute for each boolean feature).

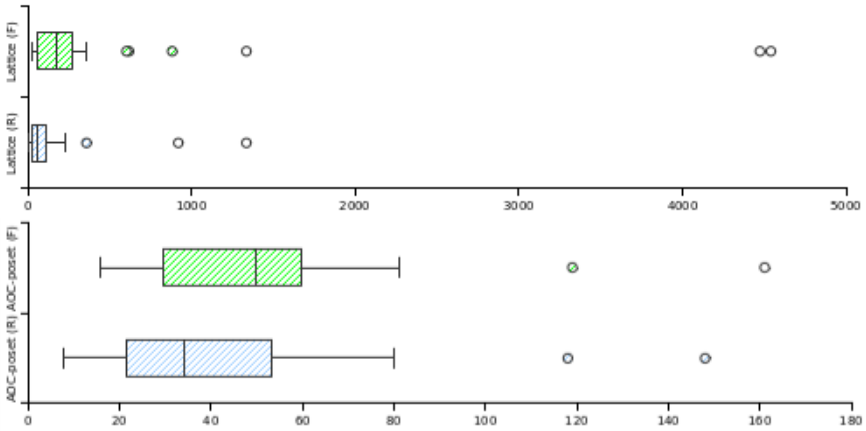
4.1 Scalability for non-connected PCMs (FCA)

Firstly, we analysed 40 PCMs from Wikipedia without taking into account relations between products. These 40 PCMs were converted into formal contexts with the method of Section 2. Results are presented in Figure 10. We have analysed 1438 products, generated 4639 binary attributes and 26002 formal concepts.

Most of the concept lattices possess between 50 and 300 concepts, but some of them can reach about 5000 concepts: this number is very high, and it would be difficult to quickly process these data. Reduced contexts (blue plots) give smaller structures, but some of them remain considerable. Thus, results of AOC-posets are encouraging: the highest number of concepts obtained is 161. Most of them possess between 30 and 60 concepts.

4.2 Scalability for connected PCMs (RCA)

Secondly, we made the same type of tests on structures which have been extended with a relation, using method of Section 3. We wanted to realise these tests on a quite important number of relationships. Yet, it is simple to find PCMs on Wikipedia but it is more difficult to automatically find relationships between these PCMs. To simulate relations between PCMs we used in the first test,



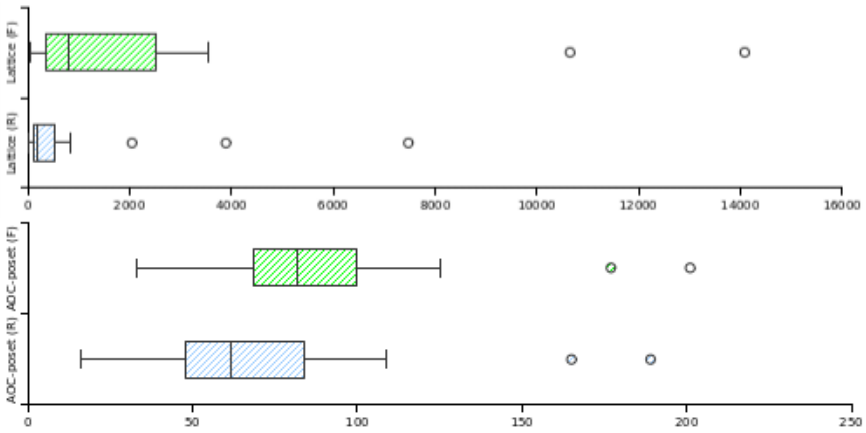
	Lattices (F)	Lattices (R)	AOC-posets (F)	AOC-posets (R)
Mean	427.375	131.35	50.3	41.025
Median	174	62.5	50	34
Minimum	21	8	16	8
Maximum	4551	1336	161	148
First Quartile	49	28.25	29.25	20.5
Third Quartile	282.5	127.75	61.25	53.75

Fig. 10. Box plots on the number of concepts obtained with concept lattices (top) and AOC-poset (bottom), from full formal contexts (green) and reduced formal contexts (blue)

we choose to automatically generate random objects-objects contexts based on two existing relations we found on Wikipedia. We analysed these relations and found out that about 75% of objects are linked to at least another object and that 95% of linked objects are linked to a single other object. We formed 20 pairs of objects-attributes contexts and generated object-objects contexts for each pair according to our analysis.

Results are presented in Figure 11. Concept lattices are huge (some can reach 14000 concepts) whereas AOC-poset remain relatively affordable (about 200 concepts).

These results match with the products used in a very simplified form for illustrating the approach. In real data, the first existing relation is between *LinuxDistribution* (77 objects, 25 features) and *FileSystem* (103 objects, 60 features). With a lattice, we obtain 1174 concepts (full context) and 1054 concepts (reduced context). With an AOC-poset, we obtain 188 then 179 concepts. The second relation is between *Wikisoftware* (43 objects, 12 features) and *Programming Language* (90 objects, 5 features). With a lattice, we obtain 282 concepts (full context) and 273 concepts (reduced context). With an AOC-poset, we obtain 87 and then 86 concepts. All the datasets (extracted from



	Lattices (F)	Lattices (R)	AOC-posets (F)	AOC-posets (R)
Mean	2263.5	878.2	89.95	72.55
Median	793	194	82	61.5
Minimum	61	24	33	16
Maximum	14083	7478	201	189
First Quartile	359.25	108.25	67.5	47.25
Third Quartile	2719.5	597.75	101.25	87.25

Fig. 11. Box plots on the number of concepts obtained with formal contexts extended with RCA

wikipedia and generated) are available for reproducibility purpose at: `http://www.lirmm.fr/recherche/equipes/marel/datasets/fca-and-pcm`.

According to these results, we can deduce that the concept lattices obtained possess a majority of empty concepts (which introduce neither attributes nor objects): AOC-poset appears like a good alternative to generate less complex structures, and keeps variability informations in the same way that concept lattices. These results on product description datasets, that are either real datasets, or datasets generated with respect to existing real one profile, allow us to think that is is realistic to envision using FCA and RCA for variability representation within a canonical form integrating features and products.

5 Related work

To our knowledge, the first research work using Formal Concept Analysis to analyse relationships between product configurations and variable features to assist construction of a product line has been realised in Loesh and Ploederer [13]. In this approach, the concept lattice is analysed to extract informations about groups of features always present, never present, always present together or never present together. Authors use these informations to describe constraints

between features and propose restructurations of these features (merge, removal or identification of alternatives feature groups). In [14], authors study, in an aspect-oriented requirements engineering context, a concept lattice which classifies scenarios by functional requirements. They analyse on the one hand the relation between concepts and quality requirements (usability, maintenance), and on the other hand interferences between quality requirements. Also, they analyse the impact of modifications. This analysis has been extended by observations about product informations contained in the lattice (or the AOC-poset), and on the manner that some work implicitly use FCA, without mentioning it [1]. In the present work, we show how to extend the original approach, which analyses products described by features, to a more general case where there are features that are products themselves. Moreover, we evaluate the scale up of FCA and RCA on product families described by PCMs from Wikipedia and linked by relationships randomly generated.

In the domain of product lines, another category of works is interested in identification of features in source code using FCA [19, 3]. In this case, described entities are variants of software systems which are described by source code and authors try to produce groups of source code elements that can be candidates to be features. Some works search for traceability links between features and the code [16]. In [7], authors cross-reference source code parts and scenarios which execute them and use features. The goal is to identify parts of the code which correspond to feature implementation. In our case, we do not work on source code, nor with scenarios, but with existing product descriptions.

Finally, a last category of works study feature organisation in FM with FCA. Some approaches [20, 15] use conceptual structures (concept lattice or AOC-poset) to recognise contraints, but also to suggest sub-features typed relations linked to the domain. In the article [15], authors study in detail the extraction of implication rules in the lattice and cover relationships, to determine for instance if a set of features covers all the products. Recent works [2, 18] focus on emphasise logical relationships between features in a FM and on the identification of transverse constraints. These logical relationships are more specifically used in [18] to analyse the variability of a set of architectural configurations. In the present work, we generate a structure or several interconnected structures. These structures are created to analyse variability, but we do not consider issues about FM construction.

6 Conclusion

In this paper, we analyse the feasibility of using Formal Concept Analysis and Concept Lattices as a complement to Product Comparison Matrices to represent variability in product lines. We propose a method to convert a description from a product comparison matrix to a concept lattice using the scaling technique. We also propose a way to model the special case where a product family is described by another product family with Relational Concept Analysis. We obtain

interconnected lattices that bring a new dimension to research and classification of products when they are in relation with other product families.

Subsequently, we realise series of tests to determine an order of magnitude of the number of concepts composing the structures obtained firstly with FCA by converting PCMs into formal contexts, and secondly with RCA by introducing relations between these contexts. In these tests, we compare two structures: concept lattices which establish a sub-order among all the concepts and AOC-posets which establish a sub-order among the concepts which introduce at least an attribute or an object. It seems that most of the concepts do not introduce any information and AOC-poset appears like a more advantageous alternative, in particular for presenting information to an end-user. Concept lattices are useful too, when they are medium-size, and for automatic data manipulations. We also show the effect of two different encoding of boolean values.

In the future, we will use the work of [4] to automatise as much as possible the conversion of PCMs. Moreover, researches on the classification process (using different scaling operators) and its applications on variability will be performed to complete this analysis. Also, a detailed study of the possibilities offered by RCA to model other cases is considered. Finally, it could be interesting to study transitions between different structures like FMs, PCMs, AOC-posets and concept lattices to be able to select one depending on the kind of operation we want to apply.

References

1. Al-Msie 'deen, R., Seriai, A.D., Huchard, M., Urtado, C., Vauttier, S., Al-Khlifat, A.: Concept lattices: A representation space to structure software variability. In: ICICS 2014: The fifth International Conference on Information and Communication Systems. pp. 1 – 6. Irbid, Jordan (Apr 2014), `http://hal-lirmm.ccsd.cnrs.fr/lirmm-01075533`
2. Al-Msie'deen, R., Huchard, M., Seriai, A., Urtado, C., Vauttier, S.: Reverse engineering feature models from software configurations using formal concept analysis. In: Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Košice, Slovakia, October 7-10, 2014. pp. 95–106 (2014)
3. Al-Msie'deen, R., Seriai, A., Huchard, M., Urtado, C., Vauttier, S., Salman, H.E.: Mining features from the object-oriented source code of a collection of software variants using formal concept analysis and latent semantic indexing. In: Proc. of The 25th SEKE. pp. 244–249 (2013)
4. Bécan, G., Sannier, N., Acher, M., Barais, O., Blouin, A., Baudry, B.: Automating the formalization of product comparison matrices. In: Proc. of the 29th ACM/IEEE ASE '14. pp. 433–444 (2014)
5. Clements, P.C., Northrop, L.M.: Software product lines: practices and patterns. Addison-Wesley (2001)
6. Czarnecki, K., Eisenecker, U.W.: Generative programming: methods, tools, and applications. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000)
7. Eisenbarth, T., Koschke, R., Simon, D.: Locating features in source code. IEEE Trans. Softw. Eng. 29(3), 210–224 (2003)

8. Ganter, B., Wille, R.: Formal concept analysis - mathematical foundations. Springer (1999)
9. Griss, M.L., Favaro, J., Alessandro, M.d.: Integrating Feature Modeling with the RSEB. In: Proceedings of the 5th International Conference on Software Reuse. pp. 76–. ICSR '98, IEEE Computer Society, Washington, DC, USA (1998), http://dl.acm.org/citation.cfm?id=551789.853486
10. Hacène, M.R., Huchard, M., Napoli, A., Valtchev, P.: Relational concept analysis: mining concept lattices from multi-relational data. Ann. Math. Artif. Intell. 67(1), 81–108 (2013)
11. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study (November 1990)
12. Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: Form: A feature-oriented reuse method with domain-specific reference architectures. Ann. Software Eng. 5, 143–168 (1998)
13. Loesch, F., Ploedereder, E.: Restructuring variability in software product lines using concept analysis of product configurations. In: Proc. of the 11th IEEE ECSMR. pp. 159–170 (2007)
14. Niu, N., Easterbrook, S.M.: Concept analysis for product line requirements. In: Proc. of the 8th AOSD 2009. pp. 137–148 (2009)
15. Ryssel, U., Ploennigs, J., Kabitzsch, K.: Extraction of feature models from formal contexts. In: Proc. of ACM SPLC '11. pp. 4:1–4:8 (2011)
16. Salman, H.E., Seriai, A., Dony, C.: Feature-to-code traceability in a collection of software variants: Combining formal concept analysis and information retrieval. In: Proc. of the14th IEEE IRI. pp. 209–216 (2013)
17. Sannier, N., Acher, M., Baudry, B.: From comparison matrix to variability model: The wikipedia case study. In: Proc. of the 28th IEEE/ACM ASE 2013. pp. 580–585 (2013)
18. Shatnawi, A., Seriai, A.D., Sahraoui, H.: Recovering architectural variability of a family of product variants. In: To appear in Proc. of the 14th ICSR (2015)
19. Xue, Y., Xing, Z., Jarzabek, S.: Feature location in a collection of product variants. In: Proc. of the 19th IEEE WCRE. pp. 145–154 (2012)
20. Yang, Y., Peng, X., Zhao, W.: Domain feature model recovery from multiple applications using data access semantics and formal concept analysis. In: Proc. of the 16th IEEE WCRE. pp. 215–224 (2009)